

LEANAGENT: LIFELONG LEARNING FOR FORMAL THEOREM PROVING

Anonymous authors

Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) have been successful in mathematical reasoning tasks such as formal theorem proving when integrated with interactive proof assistants like Lean. Existing approaches involve training or fine-tuning an LLM on a specific dataset to perform well on particular domains, such as undergraduate-level mathematics. These methods struggle with generalizability to advanced mathematics. A fundamental limitation is that these approaches operate on static domains, failing to capture how mathematicians often work across multiple domains and projects simultaneously or cyclically. We present LeanAgent, a novel lifelong learning framework for theorem proving that continuously generalizes to and improves on ever-expanding mathematical knowledge without forgetting previously learned knowledge. LeanAgent introduces several key innovations, including a curriculum learning strategy that optimizes the learning trajectory in terms of mathematical difficulty, a dynamic database for efficient management of evolving mathematical knowledge, and progressive training to balance stability and plasticity. LeanAgent successfully proves 162 theorems previously unproved by humans across 23 diverse Lean repositories, many from advanced mathematics. It performs up to $11\times$ better than the static LLM baseline, proving challenging theorems in domains like abstract algebra and algebraic topology while showcasing a clear progression of learning from basic concepts to advanced topics. In addition, we analyze LeanAgent’s superior performance on key lifelong learning metrics. LeanAgent achieves exceptional scores in stability and backward transfer, where learning new tasks improves performance on previously learned tasks. This emphasizes LeanAgent’s continuous generalizability and improvement, explaining its superior theorem proving performance.

1 INTRODUCTION

Mathematics can be expressed in informal and formal languages. Informal mathematics utilizes natural language and intuitive reasoning, whereas formal mathematics employs symbolic logic to construct machine-verifiable proofs (Kevin Buzzard, 2019). State-of-the-art large language models (LLMs), such as o1 (OpenAI, 2024) and Claude (Claude Team, 2024), produce incorrect informal proofs (Zhou et al., 2024). This highlights the importance of formal mathematics in ensuring proof correctness and reliability. Interactive theorem provers (ITPs), such as Lean (De Moura et al., 2015), have emerged as tools for formalizing and verifying mathematical proofs. However, constructing formal proofs using ITPs is complex and time-consuming; it requires extremely detailed proof steps and involves working with extensive mathematical libraries.

Recent research has explored using LLMs to generate proof steps or complete proofs. For example, LeanDojo (Yang et al., 2023) introduced the first open-source framework to spur such research. Existing approaches typically involve training or fine-tuning LLMs on a specific dataset (Jiang et al., 2022). However, data scarcity in formal theorem proving (Polu et al., 2022) hinders the generalizability of these approaches (Liu et al., 2023). For example, ReProver, the retrieval-augmented LLM from the LeanDojo family, uses a retriever fine-tuned on Lean’s math library, `mathlib4` (mathlib4 Community, 2024). Although `mathlib4` contains over 100,000 formalized mathematical theorems and definitions, it covers primarily up to undergraduate mathematics. Consequently, ReProver performs poorly on more challenging mathematics, such as Terence Tao’s formalization of the Polynomial Freiman-Ruzsa (PFR) Conjecture (Tao et al., 2024).

054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107

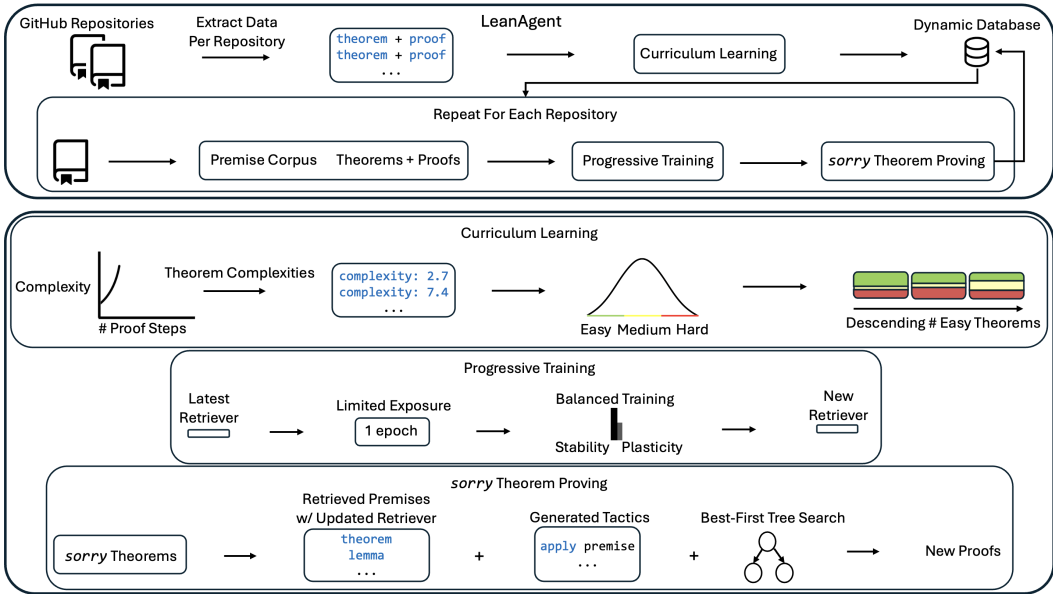


Figure 1: LeanAgent overview. LeanAgent searches for Lean repositories and uses LeanDojo to extract theorems and proofs. It uses curriculum learning, computing theorem complexity as e^S (S = proof steps), and calculating the 33rd and 67th complexity percentiles across all theorems to sort repositories by easy theorem count. LeanAgent adds the curriculum to its dynamic database. As a retrieval-based framework, LeanAgent generates a dataset (premise corpus and a collection of theorems and proofs) for each repository in the curriculum and progressively trains its retriever. Progressive training happens over one epoch to prevent forgetting old knowledge. Then, LeanAgent uses the updated retriever in a search-based method to prove results previously unproved by humans, known as *sorry* theorems. It adds new proofs to the database.

The dynamic nature of mathematical research exacerbates this generalizability issue. Mathematicians often formalize across multiple domains and projects simultaneously or cyclically. For example, Terence Tao has worked on various projects in parallel, including formalizations of the PFR Conjecture, symmetric mean of real numbers, classical Newton inequality, and asymptotic analysis (Tao; 2024a; Tao et al., 2024; Tao, 2024b). Patrick Massot has been formalizing Scholze’s condensed mathematics and the Perfectoid Spaces project (Community, 2024a;b). These examples highlight a critical gap in current theorem-proving AI approaches: the lack of a system that can adapt and improve across multiple, diverse mathematical domains over time, given limited Lean data availability.

Connection to Lifelong Learning. Crucially, how mathematicians formalize is relevant to lifelong learning, i.e. learning multiple tasks without forgetting (Wang et al., 2024b). A significant challenge is catastrophic forgetting: when adaptation to new distributions leads to a loss of understanding of old ones (Jiang et al., 2024). The core challenge is balancing plasticity (the ability to learn and adapt) with stability (the ability to retain existing knowledge) (Wang et al., 2024b). Increasing plasticity to learn new tasks efficiently can lead to overwriting previously learned information. However, enhancing stability to preserve old knowledge may impair the model’s ability to acquire new skills (van de Ven et al., 2024). Achieving the right balance is key to continuous generalizability in theorem proving.

LeanAgent. We present LeanAgent, a novel lifelong learning framework for theorem proving. As shown in Figure 1, LeanAgent’s workflow consists of (1) deriving complexity measures of theorems to compute a curriculum for learning, (2) progressive training to learn while balancing stability and plasticity, and (3) searching for proofs of *sorry* theorems (theorems unproved by humans) by leveraging a best-first tree search. LeanAgent works with any LLM; we implement it with retrieval for improved generalizability (Yang et al., 2023). LeanAgent incorporates several key innovations. It uses a custom dynamic database to manage its evolving mathematical knowledge. LeanAgent uses

108 a novel curriculum learning strategy utilizing the structure of Lean proofs to learn on increasingly
109 complex mathematical repositories.

110 We employ a simple progressive training method to avoid catastrophic forgetting. Progressive train-
111 ing allows LeanAgent to continuously adapt to new mathematical knowledge while preserving pre-
112 viously learned information. This process involves incrementally training the retriever on newly
113 generated datasets from each repository in the curriculum. Starting with a pre-trained retriever (e.g.,
114 ReProver’s retriever based on ByT5), LeanAgent trains on each new dataset for one additional epoch.
115 Restricting progressive training to one epoch helps balance stability and plasticity. Crucially, this
116 training is repeated for each dataset generated from the database, gradually expanding LeanAgent’s
117 knowledge base. This approach increases the space of possible proof states (where a state consists of
118 a theorem’s hypotheses and current proof progress) while adding new premises to the premise em-
119 beddings. More sophisticated lifelong learning methods like Elastic Weight Consolidation (EWC),
120 which uses the Fisher Information Matrix to constrain important weights for previous tasks, result
121 in excessive plasticity. The uncontrolled plasticity is due to the inability of these methods to adapt
122 parameter importance as theorem complexity increases. This forces rapid changes in parameters
123 crucial for learning advanced concepts. Such methods fail to adapt to the evolving complexity of
124 mathematical theorems, making them unsuitable for lifelong learning in theorem proving.

125 Extensive experiments across 23 diverse Lean repositories demonstrate LeanAgent’s advancements
126 in lifelong learning for theorem proving. LeanAgent successfully proves 162 theorems unproven
127 by humans, known as *sorry* theorems, many from advanced mathematics. For example, it proves
128 difficult *sorry* theorems from the PFR repository and proves challenging theorems in abstract algebra
129 and algebraic topology related to Coxeter systems and the Hairy Ball Theorem (Coxeter, 2024;
130 Hairy Ball Theorem, 2024). We find that LeanAgent demonstrates progressive learning in theorem
131 proving, initially proving basic *sorry* theorems and significantly advancing to more complex ones.
132 It outperforms the static ReProver baseline by up to $11\times$ in terms of proving new *sorry* theorems
133 while maintaining performance on known ones.

134 In theorem proving, we find that stability, without losing too much plasticity, is crucial for con-
135 tinuous generalizability to new repositories. Backward transfer (BWT), where learning new tasks
136 improves performance on previously learned tasks, is essential in theorem proving (Wang et al.,
137 2024b). Mathematicians require a lifelong learning framework for theorem proving that is both
138 *continuously generalizable* and *continuously improving*. We conduct an extensive ablation study
139 using seven lifelong metrics carefully proposed or selected from the literature. LeanAgent’s sim-
140 ple components of curriculum learning and progressive training improve stability and BWT scores
141 substantially. As such, LeanAgent earns a near-perfect composite lifelong learning score of 94%,
142 emphasizing its continuous generalizability and improvement and explaining its superior *sorry* the-
143 orem proving performance.

144 2 PRELIMINARIES

146 **Neural Theorem Proving.** The current state-of-the-art of learning-based provers employs
147 Transformer-based (Vaswani et al., 2017) LLMs that process expressions as plain text strings (Azer-
148 bayev et al., 2024; Xin et al., 2024; Shao et al., 2024). In addition, researchers have explored
149 complementary aspects like proof search algorithms (Lample et al.; Wang et al., 2023). Moreover,
150 other works break the theorem-proving process into smaller proving tasks (Song et al., 2024; Wang
151 et al., 2024a; Lin et al., 2024).

152 **Premise Selection.** A critical challenge in theorem proving is the effective selection of relevant
153 premises (Irving et al., 2016; Tworkowski et al.). However, many existing approaches treat premise
154 selection as an isolated problem (Wang & Deng, 2020; Piotrowski et al., 2023) or use selected
155 premises only as input to symbolic provers (Alama et al., 2014; Miłkowska et al., 2024).

156 **Retrieval-Augmented LLMs.** While retrieval-augmented language models have been extensively
157 studied in areas like code generation (Lu et al., 2022; Zhou et al., 2023), their application to formal
158 theorem proving is relatively new. However, relevant architectures have been researched in natural
159 language processing (NLP) (Lu et al., 2024; Borgeaud et al., 2022; Thakur et al., 2024).

160 **Lifelong Learning.** Lifelong learning addresses catastrophic forgetting in sequential task learning
161 (Chen et al., 2024). Approaches include regularization methods (Kirkpatrick et al., 2017), memory-

162 based techniques (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019; Shin et al., 2017), and knowl-
163 edge distillation (Li & Hoiem, 2017; Kim et al., 2023). Other strategies involve dynamic architec-
164 ture adjustment (Mendez & Eaton, 2021) and recent work on gradient manipulation and selective
165 re-initialization (Chen et al., 2024; Dohare et al., 2024). We justify not using these strategies in
166 Appendix A.6.

167 **Curriculum Learning in Theorem Proving.** Prior work created a synthetic inequality generator
168 to produce a curriculum of statements of increasing difficulty (Polu et al., 2022). For reinforcement
169 learning, an existing work used the length of proofs to help determine rewards (Zombori et al.,
170 2019).

171 172 173 3 METHODOLOGY 174

175
176 A useful lifelong learning strategy for theorem proving requires (a) the best repository order strategy
177 and (b) the best learning strategy. We solve (a) with curriculum learning to utilize the structure of
178 Lean proofs and (b) with progressive training to balance stability and plasticity. LeanAgent consists
179 of four main components: curriculum learning, dynamic database management, progressive training
180 of the retriever, and *sorry* theorem proving. Further methodology details are in Appendix A.1 and a
181 discussion of why curriculum learning works in theorem proving is available in Appendix A.6.

182 183 3.1 CURRICULUM LEARNING 184

185 LeanAgent uses curriculum learning to learn on increasingly complex mathematical repositories.
186 This process optimizes LeanAgent’s learning trajectory, allowing it to build upon foundational
187 knowledge before tackling more advanced concepts.

188 First, we automatically search for and clone Lean repositories from GitHub. We use LeanDojo for
189 each repository to extract fine-grained information about their theorems, proofs, and dependencies.
190 Then, we calculate the complexity of each theorem using e^S , where S represents the number of proof
191 steps. However, *sorry* theorems, which have no proofs, are assigned infinite complexity. We use an
192 exponential scaling to address the combinatorial explosion of possible proof paths as the length of
193 the proof increases. Further justification for considering this complexity metric is in Appendix A.6.

194 We compute the 33rd and 67th percentiles of complexity across all theorems in all repositories.
195 Using these percentiles, we categorize non-*sorry* theorems into three groups: easy (theorems with
196 complexity below the 33rd percentile), medium (theorems with complexity between the 33rd and
197 67th percentiles), and hard (theorems with complexity above the 67th percentile). We then sort
198 repositories by the number of easy theorems they contain. This sorting forms the basis of our
199 curriculum, with LeanAgent starting on repositories with the highest number of easy theorems.
200

201 202 3.2 DYNAMIC DATABASE MANAGEMENT 203

204 Then, we add the sorted repositories to LeanAgent’s custom dynamic database using the data Lean-
205 Agent extracted. This way, we can keep track of and interact with the knowledge that LeanAgent is
206 aware of and the proofs it has produced. We also include the complexity of each theorem computed
207 in the previous step into the dynamic database, allowing for efficient reuse of repositories in a future
208 curriculum. Details of the database’s contents and features can be found in Appendix A.1.

209 For each repository in the curriculum, LeanAgent uses the dynamic database to generate a dataset
210 by following the same procedure used to make LeanDojo Benchmark 4 (details in Appendix A.1).
211 This dataset includes a collection of theorems and their proofs. Each step of these proofs contains
212 detailed annotations, such as how the step changes the state of the proof. A state consists of a
213 theorem’s hypotheses and the current progress in proving the theorem. As such, this pairing of
214 theorems and proofs demonstrates how to use specific tactics (functions) and premises in sequence
215 to prove a theorem. In addition, the dataset includes a premise corpus, serving as a library of facts
and definitions.

3.3 PROGRESSIVE TRAINING OF THE RETRIEVER

LeanAgent then progressively trains its retriever on the newly generated dataset. This strategy allows LeanAgent to continuously adapt to new mathematical knowledge from the premises in new datasets while preserving previously learned information, crucial for lifelong learning in theorem proving. Progressive training achieves this by incrementally incorporating new knowledge from each repository.

Although LeanAgent works with any LLM, we provide a specific implementation here. We start with ReProver’s retriever, a fine-tuned version of the ByT5 encoder (Xue et al., 2022), leveraging its general pre-trained knowledge from `mathlib4`. We train LeanAgent on the new dataset for an additional epoch. This limited exposure helps prevent overfitting to the new data while allowing LeanAgent to learn essential new information. Before validation, we precompute embeddings for all premises in the corpus to ensure these embeddings are consistent with LeanAgent’s current state. To understand how LeanAgent balances stability and plasticity, we save the model iteration with the highest validation recall for the top ten retrieved premises ($R@10$). This is a raw plasticity value: it can be used to compute other metrics that describe LeanAgent’s ability to adapt to and understand new types of mathematics in the latest repository (details in Sec. 4). Then, we compute the average test $R@10$ over all previous datasets the model has progressively trained on, a raw stability value.

As mentioned previously, we repeat this procedure for each dataset we generate from the database, hence the progressive nature of this training. Progressive training adds new premises to the premise embeddings and increases the space of possible proof states. This allows LeanAgent to explore more diverse paths to prove theorems, discovering new proofs that it couldn’t produce with its original knowledge base.

3.4 *sorry* THEOREM PROVING

For each *sorry* theorem, LeanAgent generates a proof with a best-first tree search by generating tactic candidates at each step, in line with prior work (Yang et al., 2023). Using the embeddings from the entire corpus of premises we previously collected, LeanAgent retrieves relevant premises from the premise corpus based on their similarity to the current proof state, represented as a context embedding. Then, it filters the results using a corpus dependency graph to ensure that we only consider premises accessible from the current file. We add these retrieved premises to the current state and generate tactic candidates using beam search. Then, we run each tactic candidate through Lean to obtain potential next states. Each successful tactic application adds a new edge to the proof search tree. We choose the tactic with the maximum cumulative log probability of the tactics leading to it. If the search reaches a dead-end, we backtrack and explore alternative paths. We repeat the above steps until the search finds a proof, exhausts all possibilities, or reaches the time limit of 10 minutes.

If LeanAgent finds a proof, it adds it to the dynamic database. The newly added premises from this proof will be included in a future premise corpus involving the current repository. Moreover, LeanAgent can learn from the new proof during progressive training in the future, aiding further improvements.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We devote this section to describing two types of experiments: (1) *sorry* Theorem Proving: We *sorry* theorem proving performance between LeanAgent and ReProver. We also examine the progression of LeanAgent’s proven *sorry* theorems during lifelong learning to the end of lifelong learning. This shows LeanAgent’s continuous generalizability and improvement. (2) Lifelong Learning Analysis: We conduct an ablation study with seven lifelong learning metrics to explain LeanAgent’s superiority in *sorry* theorem proving. Moreover, these results explain LeanAgent’s superior handling of the stability-plasticity tradeoff. Please see Appendix A.2 for experiment implementation details.

Repositories. We evaluate our approach on a diverse set of 23 Lean repositories to assess its generalizability across different mathematical domains (Skřivan, 2024; Kontorovich, 2024; Avigad, 2024;

Table 1: Selected repository descriptions

Repository	Description
PFR	Polynomial Freiman-Ruzsa Conjecture
Hairy Ball Theorem	Algebraic topology result
Coxeter	Coxeter groups
Mathematics in Lean Source	Lean files for the textbook
Formal Book	Proofs from <i>THE BOOK</i>
MiniF2F	Math olympiad-style problem solving
SciLean	Scientific computing
Carleson	Carleson’s Theorem
Lean4 PDL	Propositional Dynamic Logic

Table 2: Theorem Proving Performance Score (TPPS) across repositories. Repositories with 0 *sorry* theorems or LeanAgent TPPS are not shown. The best TPPS for each repository is in bold.

Repository	LeanAgent	ReProver
PFR	11	1
Hairy Ball Theorem	11	1
Coxeter	11	1
PFR (alternate commit)	12	2
Mathematics in Lean Source	85	15
Formal Book	13	3
MiniF2F	225	86
SciLean	55	25
Carleson	2	2
Lean4 PDL	2	2

Tao et al., 2024; Renshaw, 2024; Fermat’s Last Theorem, 2024; DeepMind, 2024; Carneiro, 2024; Wieser, 2024; Mizuno, 2024; Murphy, 2024; Formal Logic, 2024; Con-nf, 2024; Gadgil, 2024; Yang, 2024; Zeta 3 Irrational, 2024; Firsching, Moritz, 2024; Monnerjahn, 2024; van Doorn, 2024; Dillies, 2024; Hairy Ball Theorem, 2024; Coxeter, 2024; Gattinger, 2024). Details of key repositories are in Table 1. Further details of these repositories, including commits and how we chose the initial curriculum and the sub-curriculum (described in Sec. 4.2), are in Appendix A.3.

4.2 *sorry* THEOREM PROVING

We compare the *sorry* theorems LeanAgent can prove, both during and after lifelong learning, to the ReProver baseline. We use ReProver as the baseline because we use its retriever as LeanAgent’s initial retriever in our experiments. However, we avoid percentage comparisons between LeanAgent and ReProver due to the non-linear nature of theorem proving difficulty. For example, LeanAgent’s significantly improved performance over the baseline across multiple repositories allows it to prove progressively harder theorems. Furthermore, *sorry* theorems lack ground truth proofs, so proving one is valuable for mathematical research. So, we propose a Theorem Proving Performance Score (TPPS) that emphasizes newly proven *sorry* theorems. Specifically, LeanAgent TPPS = (# ReProver Theorems Proved) + (# New Theorems Proved * X) + 1, where X represents the importance of proving a new theorem, and ReProver TPPS = (# ReProver Theorems Proved) + 1. Then, Improvement Factor = (LeanAgent TPPS)/(ReProver TPPS). We choose $X = 10$, which is relatively modest considering the large difficulty gap between basic arithmetic and abstract algebra.

In addition, a use case of LeanAgent is formalizing in a new repository after learning a curriculum; we progressively train on MiniF2F to demonstrate this. Note that we choose the Lean4 version of the MiniF2F repository (Yang, 2024) and disregard its separation into validation and test splits (reasoning in Appendix A.5). Moreover, a mathematician could use LeanAgent for (1) an initial curriculum A , and later (2) a sub-curriculum B . LeanAgent can then help the mathematician formalize the repositories in curriculum $A + B$ in parallel. To demonstrate this scenario, we continue LeanAgent on a sub-curriculum B of 8 repositories.

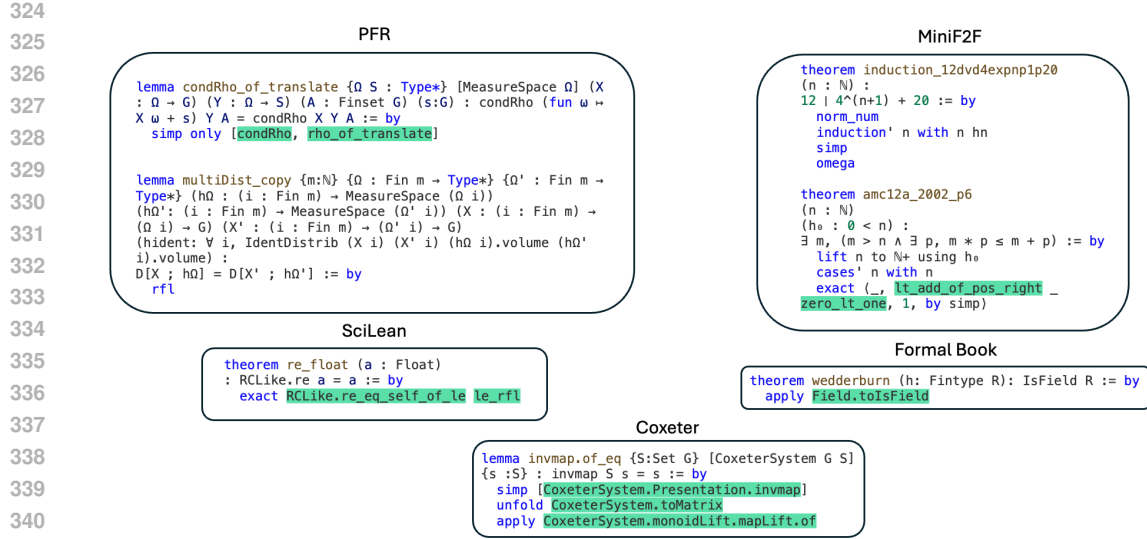


Figure 2: Case studies of LeanAgent’s new proofs. LeanAgent shows a strong understanding of these repositories, often able to retrieve the necessary premises (highlighted) in a way that represents a deep understanding of the mathematics of each repository. For example, LeanAgent proves a *sorry* theorem from PFR, `condRho_of_translate`, by simply expanding definitions, showing its deep understanding of the PFR repository. It proved a PFR theorem with just `rfl` (by reflexivity), demonstrating its understanding of how the terms in the theorem statement interrelate. In addition, LeanAgent could prove `re_float` from SciLean during lifelong learning, while ReProver could not. Moreover, its MiniF2F proofs demonstrate its ability to generate relatively longer and more complex proofs for complex mathematics. LeanAgent’s proof of `wedderburn` and `invmap.of_eq` represents a deep understanding of abstract algebra premises. Impressively, the human proof of `wedderburn` written after LeanAgent’s proof contains *175 lines*, while LeanAgent’s proof is only one line, further demonstrating a deep understanding of abstract algebra.

Results are in Table 2, with case studies in Figure 2. Appendix A.5 provides a more thorough discussion, including an ablation study, and contains the complete theorems and proofs relevant to this section.

LeanAgent demonstrates continuous generalizability and improvement in theorem-proving capabilities across multiple repositories. By the end of lifelong learning, LeanAgent has an $11\times$ improvement factor over ReProver on PFR, $5.67\times$ on Mathematics in Lean Source, $2.63\times$ on MiniF2F, and $2.2\times$ on SciLean. Additionally, LeanAgent had an $11\times$ improvement factor over ReProver on the Hairy Ball Theorem, $11\times$ on Coxeter, and $4.33\times$ on Formal Book. Its proofs are a superset of the *sorry* theorems proved by ReProver in most cases. LeanAgent progresses from basic concepts (arithmetic, simple algebra) to advanced topics (abstract algebra, topology).

PFR. LeanAgent can prove a *sorry* theorem from this cutting-edge repository, while ReProver cannot. It also generalizes to a different commit (not included in progressive training), proving a theorem with just the `rfl` tactic that ReProver cannot. Interestingly, LeanAgent has a strong enough understanding of logical manipulation within the PFR repository to prove 5 *sorry* theorems with a $0 = 1$ placeholder theorem statement (not included in TPPS) (details in Appendix A.5).

SciLean. During lifelong learning, LeanAgent proves theorems related to fundamental algebraic structures, linear and affine maps, and measure theory basics. By the end of lifelong learning, it masters concepts in advanced function spaces, sophisticated bijections, and abstract algebraic structures.

Mathematics in Lean Source. During lifelong learning, LeanAgent proves theorems about basic algebraic structures and fundamental arithmetic properties. By the end of lifelong learning, it proves complex theorems involving quantifier manipulation, set theory, and relations.

MiniF2F. ReProver demonstrates proficiency in basic arithmetic, elementary algebra, and simple calculus. However, by the end of lifelong learning, LeanAgent grasps advanced number theory, sophisticated algebra, complex calculus and analysis, abstract algebra, and complex induction.

Sub-curriculum. In the Formal Book repository, LeanAgent progresses from proving basic real analysis and number theory theorems to mastering advanced abstract algebra, exemplified by its proof of Wedderburn’s Little Theorem. For the Coxeter repository, LeanAgent proves a complex lemma about Coxeter systems, showcasing its proficiency in group theory. In the Hairy Ball Theorem repository, LeanAgent proves a key step of the theorem, demonstrating an understanding of algebraic topology. Only LeanAgent can prove these impressive theorems, demonstrating that it has much more advanced theorem-proving capabilities than ReProver.

4.3 LIFELONG LEARNING ANALYSIS

To our knowledge, no other lifelong learning frameworks for theorem proving exist in the literature. As such, we conduct an ablation study with seven lifelong learning metrics to showcase LeanAgent’s superior handling of the stability-plasticity tradeoff. These results help explain LeanAgent’s superiority in *sorry* theorem proving performance. We compute these metrics for the original curriculum of 14 repositories.

Specifically, the ablation study consists of seven additional setups constructed from a combination of learning and dataset options. Options for learning setups are progressive training with or without EWC (Kirkpatrick et al., 2017). Dataset setups involve a dataset order and construction. Options for dataset orders involve Single Repository or Merge All, where each dataset consists of all previous repositories and the new one. Given the most popular repositories on GitHub by star count, options for dataset construction include popularity order or curriculum order. Appendix A.3 shows these orders and additional repository details.

Metrics. We use seven lifelong learning metrics: Windowed-Forgetting 5 (WF5), Forgetting Measure (FM), Catastrophic Forgetting Resilience (CFR), Expanded Backward Transfer (EBWT), Windowed-Plasticity 5 (WP5), Incremental Plasticity (IP), and Composite Score (CS). A description of these metrics is in Table 3 (De Lange et al., 2023; Wang et al., 2024b; Díaz-Rodríguez et al., 2018). Our reasoning for considering these metrics is detailed in Appendix A.4.

Table 3: Description of lifelong learning metrics.

Metric	Description	Target	Type
WF5	Measures forgetting over a 5-task window	Lower	Existing
FM	Average performance drop on old tasks	Lower	Existing
CFR	Ratio of min to max average test R@10	Higher	Proposed
EBWT	Average improvement on old tasks after learning new ones	Higher	Existing
WP5	Max average test R@10 increase over a 5-task window	Higher	Existing
IP	Rate of validation R@10 change per task	Higher	Proposed
CS	Balance of stability (60%), improvement (20%), plasticity (20%)	Higher	Proposed

We describe why we introduce three new metrics to address specific aspects of lifelong learning in theorem proving:

- **Catastrophic Forgetting Resilience (CFR).** This metric captures LeanAgent’s ability to maintain performance on its weakest task relative to its best performance, crucial in the presence of diverse mathematical domains.
- **Incremental Plasticity (IP).** IP provides a more granular view of plasticity than aggregate measures and is sensitive to the order of tasks, particularly relevant in lifelong learning for theorem proving.
- **Composite Score.** To our knowledge, there isn’t a widely established composite metric that provides a single stability-plasticity trade-off score with the first six metrics in Table 3. As such, we propose a composite score: $\text{Composite Score} = 0.2 \cdot (1 - \text{WF5}_{\text{norm}}) + 0.2 \cdot (1 - \text{FM}_{\text{norm}}) + 0.1 \cdot \text{WP5}_{\text{norm}} + 0.1 \cdot \text{IP}_{\text{norm}} + 0.2 \cdot \text{EBWT}_{\text{norm}} + 0.2 \cdot \text{CFR}_{\text{norm}}$.

In addition, these metrics in the Merge All strategy measure cumulative knowledge refinement rather than isolated task performance (details in Appendix A.4). Due to these interpretational differences, we analyze Single Repository and Merge All setups separately. We consider an improvement of at least 3% to be significant.

Table 4: Comparison of lifelong learning metrics across setups. The best scores for each metric are in bold.

Metric	Single Repository				Merge All			
	LeanAgent	Setup 1	Setup 2	Setup 3	Setup 4	Setup 5	Setup 6	Setup 7
WF5	0.18	7.60	7.17	0.73	15.83	2.23	13.34	5.82
FM	0.85	6.53	4.04	2.11	10.50	4.06	11.44	3.80
CFR	0.88	0.87	0.88	0.85	0.76	0.94	0.75	0.90
EBWT	1.21	0.51	1.04	0.76	-0.20	0.73	-1.34	-0.39
WP5	2.47	0.89	1.47	3.42	0.00	0.09	0.00	0.11
IP	1.02	0.36	0.26	1.06	-1.50	-0.64	-1.71	-0.89
CS	0.94	0.16	0.47	0.61	0.16	0.97	0.04	0.78

Legend

Single Repository:	Merge All:
Setup 1: No EWC, Popularity Order	Setup 4: No EWC, Popularity Order
Setup 2: EWC, Popularity Order	Setup 5: No EWC, Curriculum Learning
Setup 3: EWC, Curriculum Learning	Setup 6: EWC, Popularity Order
	Setup 7: EWC, Curriculum Learning

Single Repository Analysis. We first analyze the Single Repository results from Table 4. LeanAgent demonstrates superior stability across multiple metrics. The WF5 metric is 75.34% lower for LeanAgent than the next best setup, suggesting it maintains performance over a window more effectively. Its FM score is 59.97% lower than Setup 3’s, showcasing its resilience against catastrophic forgetting. Furthermore, LeanAgent, Setup 1, and Setup 2 demonstrate high and consistent resilience against catastrophic forgetting, with CFR values above 0.87 and minimal (± 0.01) differences. This underscores LeanAgent’s ability to continuously generalize over time. In addition, LeanAgent has a 16.25% higher EBWT, indicating its ability to continuously improve over time.

In contrast, Setup 3 exhibits characteristics of higher plasticity. It shows a 38.26% higher WP5 over LeanAgent, indicating a greater ability to rapidly adapt to new tasks in a window. This is complemented by its 3.98% higher IP over LeanAgent, suggesting a more pronounced improvement on new tasks over time. However, these plasticity gains come at a significant cost: Setup 3 suffers from more severe catastrophic forgetting, as evidenced by its significantly worse stability metrics compared to LeanAgent. This excessive plasticity in Setup 3 stems from EWC’s inability to adapt parameter importance as theorem complexity increases. EWC preserves parameters important for simpler theorems, which may not be crucial for more complex ones. Consequently, these preserved parameters resist change while other parameters change rapidly for complex theorems. This forces the model to become more plastic overall, relying heavily on non-preserved parameters for new, complex theorems.

LeanAgent indicates superior performance in the composite score by maintaining knowledge while adapting to new tasks, making it the most suitable for lifelong learning in the Single Repository setting.

Merge All Analysis. Next, we analyze the Merge All setups from Table 4. Setup 5’s WF5 metric is 61.68% lower than the next best setup (Setup 7), suggesting Setup 5 balances and retains knowledge across an expanding dataset most effectively. Furthermore, Setup 5’s CFR score is 3.77% higher than that of Setup 7, again demonstrating high and consistent resilience in the face of an expanding, potentially more complex dataset. However, Setup 7 has a 6.44% lower FM score than Setup 5’s, showcasing its ability to maintain performance on earlier data points. Moreover, Setup 5 is the only setup with a positive EBWT, indicating that learning new tasks improves performance on the entire historical dataset. The other setups have a negative EBWT, indicating performance degradation on earlier tasks after learning new ones.

Only Setups 5 and 7 have a non-zero WP5, suggesting the ability to adapt to the growing complexity of the combined dataset. The zero values for Setups 4 and 6 indicate that popularity order struggles

486 to show improvement when dealing with merged data. However, although Setup 5 has the highest IP
 487 score with a 27.75% improvement over Setup 7, all 4 setups have negative IP values. This indicates a
 488 decrease in validation R@10 over time, suggesting that the Merge All strategy struggles to maintain
 489 performance.

490 Experiment 5’s high composite score suggests it is the best at balancing retention of earlier knowl-
 491 edge with adaptation to new data in a combined dataset. However, its negative IP value indicates a
 492 fundamental issue with its approach.

493 **Comparative Analysis and Insights.** Although the metrics have different interpretations in the Single
 494 Repository and Merge All settings, we can still draw some meaningful comparisons by focusing
 495 on overall trends and relative performance. We must consider that the negative IP values in Merge
 496 All setups indicate a significant issue. This drawback outweighs the potential benefits seen in other
 497 metrics like WP5, as it indicates a fundamental inability to maintain and improve performance in
 498 a continuously growing dataset. In contrast, LeanAgent demonstrates a positive IP, indicating its
 499 ability to incorporate new knowledge. This, combined with its superior stability and EBWT metrics
 500 relative to other Single Repository methods, suggests that LeanAgent is better suited than Setup 5
 501 for continuous generalizability and improvement.

502 **Consistency with *sorry* Theorem Proving Performance.** This lifelong learning analysis is con-
 503 sistent with LeanAgent’s *sorry* theorem proving performance. LeanAgent’s superior stability met-
 504 rics (WF5, FM, and CFR) explain its ability to maintain performance across diverse mathematical
 505 domains, as evidenced by its success in proving theorems from various repositories like SciLean,
 506 Mathematics in Lean Source, and PFR. Its high EBWT score aligns with its progression from basic
 507 concepts to advanced topics in theorem proving. While LeanAgent shows slightly lower plastic-
 508 ity (WP5 and IP) compared to some setups, this trade-off results in better overall performance,
 509 as reflected in its ability to prove a superset of *sorry* theorems compared to other methods. The
 510 composite score, consisting of continuous generalizability, continuous improvement, and plasticity,
 511 corroborates LeanAgent’s overall superiority in lifelong learning for theorem proving.

513 5 CONCLUSION

514 We have presented LeanAgent, a lifelong learning framework for theorem proving that achieves
 515 continuous generalizability and improvement across diverse mathematical domains. Key compo-
 516 nents include a curriculum learning strategy, progressive training approach, and custom dynamic
 517 database infrastructure. LeanAgent proves 162 *sorry* theorems across 23 Lean repositories, includ-
 518 ing from challenging mathematics, highlighting its potential to assist in formalizing complex proofs
 519 across multiple domains. For example, LeanAgent successfully proves advanced *sorry* theorems
 520 from the PFR repository and proves challenging theorems in abstract algebra and algebraic topol-
 521 ogy. It outperforms the ReProver baseline by up to $11\times$, progressively learning from basic to highly
 522 complex mathematical concepts. Moreover, LeanAgent shows significant performance in forgetting
 523 measures and backward transfer, achieving a near-perfect composite score of 94%. This explains its
 524 continuous generalizability and continuous improvement.

525 Future work could explore integration with Lean Copilot, providing real-time assistance with a math-
 526 ematician’s repositories. In addition, a limitation of LeanAgent is its inability to prove certain theo-
 527 rems due to a lack of data on specific topics, such as `odeSolve.arg_x0.semiAdjoint.rule`
 528 in SciLean about ODEs. To solve this problem, future work could explore using reinforcement
 529 learning for synthetic data generation during curriculum construction.

531 REFERENCES

532 Jesse Alama, Tom Heskes, Daniel Kühlwein, Evgeni Tsivtsivadze, and Josef Urban. Premise Selec-
 533 tion for Mathematics by Corpus Analysis and Kernel Methods. *Journal of Automated Reasoning*,
 534 52(2):191–213, February 2014. ISSN 0168-7433, 1573-0670. doi: 10.1007/s10817-013-9286-5.

535 Andrew Arana and Will Stafford. On the difficulty of discovering mathematical proofs. *Synthese*,
 536 202(2):1–29, 2023. doi: 10.1007/s11229-023-04184-5.

537 Jeremy Avigad. Avigad/mathematics.in_lean_source, August 2024.

- 540 Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and
541 Jeremy Avigad. ProofNet: Autoformalizing and Formally Proving Undergraduate-Level Mathe-
542 matics, February 2023.
- 543
544 Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Al-
545 bert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An Open Language Model
546 For Mathematics, March 2024.
- 547 Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning.
548 In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48,
549 Montreal Quebec Canada, June 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.
550 1553380.
- 551 Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Mil-
552 lican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego
553 de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren
554 Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol
555 Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Im-
556 proving language models by retrieving from trillions of tokens, February 2022.
- 557
558 Mario Carneiro. Digama0/lean4lean, September 2024.
- 559 Ernie Chang, Hui-Syuan Yeh, and Vera Demberg. Does the Order of Training Samples Matter? Im-
560 proving Neural Data-to-Text Generation with Curriculum Learning. In Paola Merlo, Jorg Tiede-
561 mann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of*
562 *the Association for Computational Linguistics: Main Volume*, pp. 727–733, Online, April 2021.
563 Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.61.
- 564
565 Arslan Chaudhry, Marc Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient
566 Lifelong Learning with A-GEM, January 2019.
- 567 Jiefeng Chen, Timothy Nguyen, Dilan Gorur, and Arslan Chaudhry. Is forgetting less a good induc-
568 tive bias for forward transfer?, March 2023.
- 569
570 Yupeng Chen, Senmiao Wang, Zhihang Lin, Zeyu Qin, Yushun Zhang, Tian Ding, and Ruoyu Sun.
571 MoFO: Momentum-Filtered Optimizer for Mitigating Forgetting in LLM Fine-Tuning. 2024. doi:
572 10.48550/ARXIV.2407.20999.
- 573 Volkan Cirik, Eduard Hovy, and Louis-Philippe Morency. Visualizing and Understanding Curricu-
574 lum Learning for Long Short-Term Memory Networks, November 2016.
- 575
576 Claude Team. Introducing claude 3.5 sonnet, June 2024. URL <https://www.anthropic.com/news/claude-3-5-sonnet>.
- 577
578 Lean Community. Leanprover-community/lean-liquid. leanprover-community, September 2024a.
- 579
580 Lean Community. Leanprover-community/lean-perfectoid-spaces. leanprover-community, August
581 2024b.
- 582
583 Con-nf. Leanprover-community/con-nf: A formal consistency proof of Quine’s set theory New
584 Foundations. <https://github.com/leanprover-community/con-nf/tree/main>, 2024.
- 585
586 Coxeter. NUS-Math-Formalization/coxeter at 96af8aee7943ca8685ed1b00cc83a559ea389a97.
587 <https://github.com/NUS-Math-Formalization/coxeter/tree/96af8aee7943ca8685ed1b00cc83a559ea389a97>,
2024.
- 588
589 Matthias De Lange, Gido van de Ven, and Tinne Tuytelaars. Continual evaluation for lifelong
590 learning: Identifying the stability gap, March 2023.
- 591
592 Leonardo De Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob Von Raumer. The
593 Lean Theorem Prover (System Description). In Amy P. Felty and Aart Middeldorp (eds.), *Automated Deduction - CADE-25*, volume 9195, pp. 378–388, Cham, 2015. Springer International Publishing. ISBN 978-3-319-21400-9 978-3-319-21401-6. doi: 10.1007/978-3-319-21401-6_26.

- 594 Google DeepMind. Google-deepmind/debate. Google DeepMind, August 2024.
595
- 596 Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget,
597 there is more than forgetting: New metrics for Continual Learning, October 2018.
- 598 Yaël Dillies. YaelDillies/LeanAPAP, September 2024.
599
- 600 Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mah-
601 mood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026):
602 768–774, August 2024. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-024-07711-7.
- 603 Fermat's Last Theorem. ImperialCollegeLondon/FLT. Imperial College London, September 2024.
604
- 605 Firsching, Moritz. Mo271/FormalBook: Formalizing "Proofs from THE BOOK", 2024.
606
- 607 Formalized Formal Logic. FormalizedFormalLogic/Foundation. FormalizedFormalLogic, Septem-
608 ber 2024.
- 609 Siddhartha Gadgil. Siddhartha-gadgil/Saturn: Experiments with SAT solvers with proofs in Lean 4.
610 <https://github.com/siddhartha-gadgil/Saturn>, 2024.
- 611 Malvin Gattinger. M4lvin/lean4-pdl, September 2024.
612
- 613 Hairy Ball Theorem. Corent1234/hairy-ball-theorem-lean at
614 a778826d19c8a7ddf1d26beeea628c45450612e6. [https://github.com/corent1234/hairy-ball-](https://github.com/corent1234/hairy-ball-theorem-lean/tree/a778826d19c8a7ddf1d26beeea628c45450612e6)
615 [theorem-lean/tree/a778826d19c8a7ddf1d26beeea628c45450612e6](https://github.com/corent1234/hairy-ball-theorem-lean/tree/a778826d19c8a7ddf1d26beeea628c45450612e6), 2024.
- 616 Geoffrey Irving, Christian Szegedy, Alexander A Alemi, Niklas Een, Francois Chollet, and Josef
617 Urban. DeepMath - Deep Sequence Models for Premise Selection. In *Advances in Neural Infor-*
618 *mation Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- 619 Albert Q. Jiang, Wenda Li, Szymon Tworowski, Konrad Czechowski, Tomasz Odrzygóźdź, Piotr
620 Miłoś, Yuhuai Wu, and Mateja Jamnik. Thor: Wielding Hammers to Integrate Language Models
621 and Automated Theorem Provers. 2022. doi: 10.48550/ARXIV.2205.10893.
- 622 Gangwei Jiang, Caigao Jiang, Zhaoyi Li, Siqiao Xue, Jun Zhou, Linqi Song, Defu Lian, and Ying
623 Wei. Interpretable Catastrophic Forgetting of Large Language Model Fine-tuning via Instruction
624 Vector. 2024. doi: 10.48550/ARXIV.2406.12227.
- 625 Kevin Buzzard. The Future of Mathematics? Professor Kevin Buzzard - 30 May 2019, June 2019.
626
- 627 Seungyeon Kim, Ankit Singh Rawat, Manzil Zaheer, Sadeep Jayasumana, Veeranjaneyulu Sad-
628 hanala, Wittawat Jitkrittum, Aditya Krishna Menon, Rob Fergus, and Sanjiv Kumar. EmbedDis-
629 till: A Geometric Knowledge Distillation for Information Retrieval. 2023. doi: 10.48550/ARXIV.
630 2301.12005.
- 631 James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A.
632 Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hass-
633 abis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting
634 in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March
635 2017. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1611835114.
- 636 Alex Kontorovich. AlexKontorovich/PrimeNumberTheoremAnd, August 2024.
637
- 638 Guillaume Lample, Marie-Anne Lachaux, Thibaut Lavril, Xavier Martinet, Amaury Hayat, Gabriel
639 Ebner, Aurélien Rodriguez, and Timothée Lacroix. HyperTree Proof Search for Neural Theorem
640 Proving.
641
- 642 Conglong Li, Zhewei Yao, Xiaoxia Wu, Minjia Zhang, Connor Holmes, Cheng Li, and Yuxiong He.
643 DeepSpeed Data Efficiency: Improving Deep Learning Model Quality and Training Efficiency
644 via Efficient Data Sampling and Routing. *Proceedings of the AAAI Conference on Artificial*
645 *Intelligence*, 38(16):18490–18498, March 2024. ISSN 2374-3468, 2159-5399. doi: 10.1609/
646 [aaai.v38i16.29810](https://doi.org/10.1609/aaai.v38i16.29810).
- 647 Zhizhong Li and Derek Hoiem. Learning without Forgetting, February 2017.

- 648 Haohan Lin, Zhiqing Sun, Yiming Yang, and Sean Welleck. Lean-STaR: Learning to Interleave
649 Thinking and Proving, August 2024.
- 650
- 651 Chengwu Liu, Jianhao Shen, Huajian Xin, Zhengying Liu, Ye Yuan, Haiming Wang, Wei Ju,
652 Chuanyang Zheng, Yichun Yin, Lin Li, Ming Zhang, and Qun Liu. FIMO: A Challenge For-
653 mal Dataset for Automated Theorem Proving, December 2023.
- 654 David Lopez-Paz and Marc’ Aurelio Ranzato. Gradient Episodic Memory for Continual Learning.
655 In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- 656
- 657 I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. In *International Conference*
658 *on Learning Representations*, November 2017.
- 659 Minghai Lu, Benjamin Delaware, and Tianyi Zhang. Proof Automation with Large Language Mod-
660 els, September 2024.
- 661
- 662 Shuai Lu, Nan Duan, Hojae Han, Daya Guo, Seung-won Hwang, and Alexey Svyatkovskiy. ReACC:
663 A Retrieval-Augmented Code Completion Framework. In Smaranda Muresan, Preslav Nakov,
664 and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for*
665 *Computational Linguistics (Volume 1: Long Papers)*, pp. 6227–6240, Dublin, Ireland, May 2022.
666 Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.431.
- 667 The mathlib4 Community. Leanprover-community/mathlib4. leanprover-community, September
668 2024.
- 669
- 670 Jorge A Mendez and Eric Eaton. Lifelong Learning of Compositional Structures. 2021.
- 671 Maciej Mikula, Szymon Tworkowski, Szymon Antoniak, Bartosz Piotrowski, Albert Qiaochu Jiang,
672 Jin Peng Zhou, Christian Szegedy, Łukasz Kuciński, Piotr Miłoś, and Yuhuai Wu. Magnusham-
673 mer: A Transformer-Based Approach to Premise Selection, March 2024.
- 674
- 675 Yuma Mizuno. Yuma-mizuno/lean-math-workshop. [https://github.com/yuma-mizuno/lean-math-](https://github.com/yuma-mizuno/lean-math-workshop)
676 [workshop](https://github.com/yuma-mizuno/lean-math-workshop), 2024.
- 677 Ludwig Monnerjahn. Louis-Le-Grand/Formalisation-of-constructable-numbers, September 2024.
- 678
- 679 Logan Murphy. Loganrjmurphy/LeanEuclid, September 2024.
- 680
- 681 OpenAI. OpenAI o1 System Card, September 2024.
- 682 Bartosz Piotrowski, Ramon Fernández Mir, and Edward Ayers. Machine-Learned Premise Selec-
683 tion for Lean. In *Automated Reasoning with Analytic Tableaux and Related Methods: 32nd*
684 *International Conference, TABLEUX 2023, Prague, Czech Republic, September 18–21, 2023,*
685 *Proceedings*, pp. 175–186, Berlin, Heidelberg, September 2023. Springer-Verlag. ISBN 978-3-
686 031-43512-6. doi: 10.1007/978-3-031-43513-3_10.
- 687 Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and
688 I. Sutskever. Formal Mathematics Statement Curriculum Learning. *ArXiv*, February 2022.
- 689
- 690 David Renshaw. Dwrensha/compfiles: Catalog Of Math Problems Formalized In Lean.
691 <https://github.com/dwrensha/compfiles>, September 2024.
- 692
- 693 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,
694 Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of
695 Mathematical Reasoning in Open Language Models, April 2024.
- 696
- 697 Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Gener-
ative Replay, December 2017.
- 698
- 699 Tomáš Skřivan. Lecopivo/SciLean: Scientific computing in Lean 4.
700 <https://github.com/lecopivo/SciLean/tree/master>, September 2024.
- 701 Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Towards large language models as copilots
for theorem proving in lean, 2024. URL <https://arxiv.org/abs/2404.12534>.

- 702 Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Baby Steps: How “Less is More” in
703 Unsupervised Dependency Parsing.
704
- 705 Sandeep Subramanian, Sai Rajeswar, Francis Dutil, Chris Pal, and Aaron Courville. Adversarial
706 Generation of Natural Language. In Phil Blunsom, Antoine Bordes, Kyunghyun Cho, Shay
707 Cohen, Chris Dyer, Edward Grefenstette, Karl Moritz Hermann, Laura Rimell, Jason Weston,
708 and Scott Yih (eds.), *Proceedings of the 2nd Workshop on Representation Learning for NLP*,
709 pp. 241–251, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi:
710 10.18653/v1/W17-2629.
- 711 Terence Tao. Teorth/asymptotic.
712
- 713 Terence Tao. Teorth/newton, June 2024a.
714
- 715 Terence Tao. Teorth/symmetric_project, July 2024b.
716
- 717 Terence Tao, Pietro Monticone, Lorenzo Lucciolli, and Rémy Degenne. Teorth/pfr, August 2024.
718
- 719 Amitayush Thakur, George Tsoukalas, Yeming Wen, Jimmy Xin, and Swarat Chaudhuri. An In-
720 Context Learning Agent for Formal Theorem-Proving, August 2024.
- 721 George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Ami-
722 tayush Thakur, and Swarat Chaudhuri. PutnamBench: Evaluating Neural Theorem-Provers on
723 the Putnam Mathematical Competition, July 2024.
- 724 Szymon Tworkowski, Maciej Mikula, Tomasz Odrzygózdz, Konrad Czechowski, Szymon Antoniak,
725 Albert Q Jiang, Christian Szegedy, Lukasz Kucinski, Piotr Milos, and Yuhuai Wu. Formal Premise
726 Selection With Language Models.
727
- 728 Guido M. van de Ven, Nicholas Soures, and Dhireesha Kudithipudi. Continual Learning and Cata-
729 strophic Forgetting, March 2024.
- 730 Floris van Doorn. Fpvandoorn/carleson, September 2024.
- 731 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez,
732 Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural In-
733 formation Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- 734 Haiming Wang, Ye Yuan, Zhengying Liu, Jianhao Shen, Yichun Yin, Jing Xiong, Enze Xie, Han
735 Shi, Yujun Li, Lin Li, Jian Yin, Zhenguo Li, and Xiaodan Liang. DT-Solver: Automated Theorem
736 Proving with Dynamic-Tree Sampling Guided by Proof-level Value Function. In Anna Rogers,
737 Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the
738 Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12632–12646, Toronto,
739 Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.
740 706.
- 741 Haiming Wang, Huajian Xin, Zhengying Liu, Wenda Li, Yinya Huang, Jianqiao Lu, Zhicheng Yang,
742 Jing Tang, Jian Yin, Zhenguo Li, and Xiaodan Liang. Proving Theorems Recursively, May 2024a.
743
- 744 Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A Comprehensive Survey of Continual
745 Learning: Theory, Method and Application. *IEEE Transactions on Pattern Analysis and Machine
746 Intelligence*, 46(8):5362–5383, August 2024b. ISSN 0162-8828, 2160-9292, 1939-3539. doi:
747 10.1109/TPAMI.2024.3367329.
- 748 Mingzhe Wang and Jia Deng. Learning to Prove Theorems by Learning to Generate Theorems.
749 *ArXiv*, February 2020.
- 750 Eric Wieser. Eric-wieser/lean-matrix-cookbook: The matrix cookbook, proved in the Lean theorem
751 prover. <https://github.com/eric-wieser/lean-matrix-cookbook>, 2024.
752
- 753 Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanxia Zhao, Haocheng Wang, Bo Liu,
754 Liyue Zhang, Xuan Lu, Qishi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu,
755 Fuli Luo, and Chong Ruan. DeepSeek-Prover-V1.5: Harnessing Proof Assistant Feedback for
Reinforcement Learning and Monte-Carlo Tree Search, August 2024.

- 756 Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam
757 Roberts, and Colin Raffel. ByT5: Towards a Token-Free Future with Pre-trained Byte-to-Byte
758 Models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022. doi:
759 10.1162/tacl.a.00461.
- 760 Kaiyu Yang. Yangky11/miniF2F-lean4. <https://github.com/yangky11/miniF2F-lean4/tree/main>,
761 2024.
- 762 Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil,
763 Ryan Prenger, and Anima Anandkumar. LeanDojo: Theorem Proving with Retrieval-Augmented
764 Language Models. 2023. doi: 10.48550/ARXIV.2306.15626.
- 765
766 Wojciech Zaremba and Ilya Sutskever. Learning to Execute, February 2015.
- 767
768 Zeta 3 Irrational. Ahhwuhu/zeta_3_irrational at 3d68ddd90434a398c9a72f30d50c57f15a0118c7.
769 https://github.com/ahhwuhu/zeta_3_irrational/tree/3d68ddd90434a398c9a72f30d50c57f15a0118c7,
770 2024.
- 771 Jin Peng Zhou, Charles Staats, Wenda Li, Christian Szegedy, Kilian Q. Weinberger, and Yuhuai Wu.
772 Don’t Trust: Verify – Grounding LLM Quantitative Reasoning with Autoformalization, March
773 2024.
- 774
775 Shuyan Zhou, Uri Alon, Frank F. Xu, Zhiruo Wang, Zhengbao Jiang, and Graham Neubig.
776 DocPrompting: Generating Code by Retrieving the Docs, February 2023.
- 777
778 Zsolt Zombori, Adrian Csizsarik, Henryk Michalewski, Cezary Kaliszyk, and Josef Urban. Curricu-
779 lum Learning and Theorem Proving. 2019.

781 A APPENDIX

782 A.1 FURTHER METHODOLOGY DETAILS

783
784 **Repository Scanning and Data Extraction.** We use the GitHub API to query for Lean repositories
785 based on sorting parameters (e.g., by repository stars or most recently updated repositories). We
786 maintain a list of known repositories to avoid; the list can be updated to allow LeanAgent to re-
787 analyze the same repository on a new commit or Lean version.

788
789 We clone each identified repository locally using the Git version control system. To ensure compati-
790 bility with our theorem-proving pipeline, we check the Lean version required by each repository and
791 compare it with the supported versions of our system. If the required version is incompatible, we
792 skip the repository and move on to the next one. Otherwise, LeanAgent switches its Lean version
793 to match the repository’s version. This version checking is performed by parsing the repository’s
794 configuration files and extracting the specified Lean version.

795
796 **Dynamic Database Management.** This database contains many key features that are useful in our
797 setting. For example, it can add new repositories, update existing ones, and generate merged datasets
798 from multiple repositories with customizable splitting strategies. In addition, it can query specific
799 theorems or premises across repositories, track the progress of proof attempts (including the proof
800 status of *sorry* theorems), and analyze the structure and content of Lean proofs, including tactic
sequences and proof states.

801
802 The database keeps track of various details: Repository metadata; theorems categorized as already
803 proven, *sorry* theorems that are proven, or *sorry* theorems that are unproven; premise files with their
804 imports and individual premises; traced files for tracking which files have been processed; detailed
805 theorem information, including file path, start/end positions, and full statements; and traced tactics
with annotated versions, including the proof state before and after application.

806
807 If we encounter duplicate theorems between repositories while merging repositories, we use the
808 theorem from the repository most recently added to the database. We deduplicate premise files and
809 traced files by choosing the first one encountered while merging the repositories. We also generate
metadata containing details of all the repositories used to generate the dataset and statistics regarding
the theorems, premise files, and traced files in the dataset, such as the total number of theorems.

We provide the user with many options to generate a dataset. To generate the set of theorems and proofs, the default option is to simply use the theorems, proofs, premise files, and traced files from the current curriculum repository in the database. Specifically, we use the random split from LeanDojo to create training, validation, and testing sets. We refrain from using the novel split from LeanDojo, as we would like LeanAgent to learn as much as possible from a repository to perform well on its hardest theorems. The data in the splits include details about the proofs of theorems, including the URL and commit of the source repository, the file path of the theorem, the full name of the theorem, the theorem statement, the start and end positions in the source file, and a list of traced tactics with annotations. The validation and test split each contain 2% of the total theorem and proofs, following the methodology from LeanDojo. Moreover, the database uses a topological sort over the traced files in the repository to generate the premise corpus. This corpus is a JSON Lines file, where each line is a JSON object consisting of a path to a Lean source file, the file’s imports, and the file’s premise statements and definitions.

Progressive Training of the Retriever. We describe some additional steps for progressive training. To precompute the embeddings, we use a single forward pass with batch processing to serialize and tokenize premises from the entire corpus. Then, we use the retriever’s encoder to process the batches and generate embeddings.

sorry Theorem Proving. We start by processing the premise corpus to use it more efficiently during premise retrieval. This involves initializing a directed dependency graph to represent each file path in the corpus, adding files as nodes and imports as edges, and creating a transitive closure of this graph. We also track all premises encountered during this process, building a comprehensive knowledge base.

Crucially, we limit retrieval to a subset of all available premises to aid the effectiveness of the results. Specifically, we choose the top 25% of accessible and relevant premises, following ReProver’s method.

Proof Integration and Pull Request Generation. We integrate the generated proofs into the original Lean files and create pull requests to propose the changes to the repository owners. This aids the development of these repositories and functions as more training data for future research.

To achieve this, in a temporary Git branch, we iterate over the Lean files and locate the *sorry* keywords corresponding to the generated proofs. We then replace these *sorry* keywords with the actual proof text, working from the bottom of each file upward to preserve the position of theorems. After integrating the proofs, we commit our changes, push them, and create a pull request for the repository on GitHub.

A.2 EXPERIMENT IMPLEMENTATION DETAILS

We use ReProver’s retriever trained on the random split from LeanDojo. We use four NVIDIA A100 GPUs with 80GB of memory each for progressive training. LeanAgent uses a distributed architecture leveraging PyTorch Lightning and Ray for parallel processing. We use bfloat16 mixed precision and optimize with AdamW (Loshchilov & Hutter, 2017) with an effective batch size of 16 (achieved through a batch size of 4 with gradient accumulation over 4 steps). In the first 1,000 steps, the learning rate warms up linearly from 0 to the maximum value of 10^{-3} . Then it decays to 0 using a cosine schedule. In addition, we apply gradient clipping with a value of 1.0. Just as ReProver does during training, we sample 3 negative premises per example, including 1 in-file negative premise. The maximum sequence length for the retriever is set to 1024 tokens. The maximum sequence length for the generator is set to 512 tokens for input and 128 tokens for output.

The prover uses a best-first search strategy with no limit on the maximum number of expansions of the search tree. It generates 64 tactic candidates and retrieves 100 premises for each proof state. LeanAgent uses ReProver’s tactic generator for the experiments. We generate tactics with a beam search of size 5. We used 4 CPU workers, 1 per GPU. Due to the wide variety of repositories and experimental setups that we tested, the time for each experiment varied from 4 to 9 days to complete.

Furthermore, we do not compare LeanAgent with any existing LLM-based prover besides ReProver because LeanAgent is a framework, not a model. As mentioned previously, it can be used with any LLM. As such, a comparison would be impractical for reasons including differences in data, pre-

training, and fine-tuning. We only compare with ReProver because we use ReProver’s retriever as the starting one in LeanAgent, allowing for a more faithful comparison.

Moreover, the objective function for Elastic Weight Consolidation (EWC) is given by:

$$L(\theta) = L_B(\theta) + \frac{\lambda}{2} \sum_i F_i(\theta_i - \theta_{A,i})^2$$

where $L_B(\theta)$ is the loss for the current task B, i is the label for each parameter, $\theta_{A,i}$ are the parameters from the previous task A, F_i is the Fisher information matrix, and λ is a hyperparameter that controls the strength of the EWC penalty. For the setups that use EWC, we performed a grid search over λ values in $\{0.01, 0.1, 1, 10, 100\}$. For each value, we ran Setup 2 on separate testing repositories. We found 0.1 to yield the best overall composite score.

A.3 REPOSITORY DETAILS

Table 5: Additional repository descriptions

Repository	Description
Prime Number Theorem And	Prime Number Theorem proof
Compfiles	Catalog of Olympiad-style math problems
FLT	Fermat’s Last Theorem proof
Debate	Stochastic double-efficient debate protocol
Lean4Lean	Implementation of Lean4 kernel in Lean4
Matrix Cookbook	The Matrix Cookbook lemmas
Matrix Workshop	Detailed Lean tutorial
LeanEuclid	Euclidean Geometry
Foundation	Formal logic results
Con-nf	Consistency of Quine’s New Foundations
Saturn	SAT solver-prover implementation
Zeta 3 Irrational	Proof of $\zeta(3)$ irrationality
Formalization of Constructable Numbers	Ancient construction problems
LeanAPAP	Kelley-Meka bound on Roth numbers

Repository Statistics and Information. Additional repository descriptions are in Table 5. The commits we used for experiments are in Table 6. Moreover, the repository orders are detailed in Table 7 and Table 8.

Repository Selection Process. Many repositories have issues such as incompatibilities with LeanDojo, unsupported Lean versions, and build failures. As such, our process for choosing the 14 repositories in the first curriculum was simply using LeanAgent to extract information from the most popular repositories on GitHub. We disregard incompatible and inapplicable ones, such as those with no theorems. We performed this process on August 20, 2024. We performed a similar process for the eight repositories in the second curriculum with two differences: (1) We checked that the number of *sorry* theorems visible from GitHub was at least 10. This narrowed down the available repositories significantly. (2) We included some more recently updated repositories to provide some variety in the age of the repositories in our curriculum. We performed this process on September 14, 2024. However, many of the repositories that passed this test had fewer than 10 *sorry* theorems when processed by LeanDojo; this is mainly due to the functionalities of LeanDojo.

Also, given the inherent need of premises for progressive training to succeed, we do not apply LeanAgent on repositories such as ProofNet (Azerbaiyev et al., 2023) and PutnamBench (Tsoukalas et al., 2024) that have no premises.

A.4 LIFELONG LEARNING METRIC DETAILS

Prior work has noted that lifelong learning methods generally lack standard evaluation metrics (De Lange et al., 2023; Díaz-Rodríguez et al., 2018). As such, our selection primarily focused

Table 6: Repository commits. Formalization of Const. Numbers denotes Formalization of Constructable Numbers.

Repository	Commit
PFR	fa398a5b853c7e94e3294c45e50c6aee013a2687
Hairy Ball Theorem	a778826d19c8a7ddf1d26beeea628c45450612e6
Coxeter	96af8aee7943ca8685ed1b00cc83a559ea389a97
Mathematics in Lean Source	5297e0fb051367c48c0a084411853a576389ecf5
Formal Book	6fbe8c2985008c0bfb30050750a71b90388ad3a3
MiniF2F	9e445f5435407f014b88b44a98436d50dd7abd00
SciLean	22d53b2f4e3db2a172e71da6eb9c916e62655744
Carleson	bec7808b907190882fa1fa54ce749af297c6cf37
Lean4 PDL	c7f649fe3c4891cf1a01c120e82ebc5f6199856e
Prime Number Theorem And	29badd685660b5fedd7bd67f9916ae24253d566
Compfiles	f99bf6f2928d47dd1a445b414b3a723c2665f091
FLT	b208a302cdbcfadce33d8165f0b054bfa17e2147
Debate	7fb39251b705797ee54e08c96177fabd29a5b5a3
Lean4Lean	05b1f4a68c5facea96a5ee51c6a56fef21276e0f
Matrix Cookbook	f15a149d321ac99ff9b9c024b58e7882f564669f
Matrix Workshop	5acd4b933d47fd6c1032798a6046c1baf261445d
LeanEuclid	f1912c3090eb82820575758efc31e40b9db86bb8
Foundation	d5fe5d057a90a0703a745cdc318a1b6621490c21
Con-nf	00bdc85ba7d486a9e544a0806a1018dd06fa3856
Saturn	3811a9dd46cdfd5fa0c0c1896720c28d2ec4a42a
Zeta 3 Irrational	914712200e463cfc97fe37e929d518dd58806a38
Formalization of Const. Numbers	01ef1f22a04f2ba8081c5fb29413f515a0e52878
LeanAPAP	951c660a8d7ba8e39f906fdf657674a984effa8b

Table 7: Repository orders (initial curriculum). Note that Popularity Order is by star count on August 20, 2024.

#	Curriculum Order	Popularity Order
1	Compfiles	SciLean
2	Mathematics in Lean Source	FLT
3	Prime Number Theorem And	PFR
4	Math Workshop	Prime Number Theorem And
5	FLT	Compfiles
6	PFR	Debate
7	SciLean	Mathematics in Lean Source
8	Debate	Lean4Lean
9	Matrix Cookbook	Matrix Cookbook
10	Con-nf	Math Workshop
11	Foundation	LeanEuclid
12	Saturn	Foundation
13	LeanEuclid	Con-nf
14	Lean4Lean	Saturn

Table 8: Curriculum order (sub-curriculum)

#	Curriculum Order
1	Zeta 3 Irrational
2	Formal Book
3	Formalization of Constructable Numbers
4	Carleson
5	LeanAPAP
6	Hairy Ball Theorem
7	Coxeter
8	Lean4 PDL

on metrics that emphasized a change over time, aligning with our problem setup. In addition, we removed metrics that were redundant. For example, prior work suggests that evaluating lifelong learning frameworks only after each task, rather than over time, leads to substantial forgetting (De Lange et al., 2023). As such, we adopt WF and WP in our analysis of LeanAgent. We use a window size of 5 for WF and WP as this represents a relatively medium-term understanding, given that we have 14 repositories. This would provide a balanced interpretation of forgetting and plasticity. Furthermore, we use the EBWT metric, in line with previous work, to evaluate LeanAgent throughout its lifetime rather than simply at the end (Díaz-Rodríguez et al., 2018). Moreover, we chose not to include the Forward Transfer metric as prior work has shown that a lower FM leads to better forward transfer (Chen et al., 2023). As such, we only check FM. We also chose not to include lifelong learning metrics for overall performance, such as Time Weighted Cumulative Performance (TWCP), Area Under the Learning Curve (AULC), and Average Accuracy (AA), as these would lead to redundancy in our analysis. Specifically, the metrics we chose were all computed using validation R@10 and the average test R@10, which are already measures of LeanAgent’s performance.

We provide some additional details on the metrics we used. Windowed-Forgetting 5 (WF5) quantifies model stability by measuring the maximum performance decrease in average test R@10 over a sliding window of 5 evaluations. Following prior work, we define WF for a given window size and then average it over all evaluation tasks to provide a single measure of stability. Moreover, Catastrophic Forgetting Resilience (CFR) is a key indicator of the stability-plasticity trade-off. Furthermore, the Forgetting Measure (FM) measures the negative influence that learning a task has on the test R@10 of all old tasks. It is the average forgetting of all old tasks, where forgetting of a task is the difference between its highest and current performance. Furthermore, BWT measures the positive influence of learning a new task on the test R@10 of old tasks. EBWT improves upon this metric by considering the average of the BWT computed after each task. Windowed-Plasticity 5 (WP5) measures the ability to learn new information by quantifying the maximum average test R@10 increase over a sliding window of 5 evaluations. Incremental Plasticity (IP) tracks changes in validation R@10 for each task over time.

However, it is important to note that our lifelong learning metrics have different interpretations in the Merge All dataset construction strategy, which differs from the traditional task-incremental setup. To our knowledge, an interpretation of these metrics in this setting has not been thoroughly conducted. As such, we propose that metrics should be interpreted with an understanding that they may reflect an adaptation to gradual shifts in data distribution rather than abrupt task changes. Specifically, WF5 may reflect not just forgetting old tasks but also the ability to balance and retain knowledge across an expanding dataset. WP5 could indicate how well the model adapts to the growing complexity of the combined dataset rather than purely learning new, isolated tasks. FM, in this context, may represent the ability to maintain performance on earlier data points as the dataset grows. EBWT might reflect the capacity to leverage newly added data to improve performance on the entire historical dataset. CFR becomes a measure of stability in the face of an expanding, potentially more complex dataset. IP may represent how quickly the model adapts to the evolving nature of the combined dataset rather than discrete new tasks. The composite score in this context reflects the ability to handle an expanding, more complex dataset. These metrics in the Merge All case measure the ability to accumulate and refine knowledge over time rather than strictly measuring performance on isolated tasks.

It is worth analyzing the effect of EWC. Our results in Sec. 4.3 suggest that the effect of EWC is not uniform across different task-ordering strategies. In curriculum-based ordering, EWC seems to improve plasticity (WP5 and IP) at the cost of stability and continuous improvement (WF5, FM, EBWT, and CFR). An exception is Setup 7, which improves WP5 and FM. This suggests that the Merge All strategy creates a more nuanced balance between stability and plasticity. EWC generally improves stability and plasticity metrics, except IP, in the popularity order for the Single Repository strategy. This may be because this ordering is less optimized for learning, and EWC helps to mitigate some of its shortcomings. However, when used with a more effective curriculum-based ordering, EWC interferes with the carefully structured learning process, leading to mixed results. Moreover, in the Merge All scenario, EWC offers benefits only for the curriculum learning setups, suggesting that its effectiveness might be limited in more complex, merged datasets. This can be explained by the fact that the Merge All strategy is a memory-based approach to lifelong learning. As such, using both EWC and the Merge All strategy may lead to excessive stability. This analysis further explains why LeanAgent’s setup is superior.

1026 Table 9: *sorry* proofs across repositories and setups. Total describes the total number of *sorry*
 1027 theorems in each repository, and the remaining columns describe the number of *sorry* theorems
 1028 that each setup could prove. MIL denotes the Mathematics in Lean Source repository. The highest
 1029 number of theorems proved is in bold.

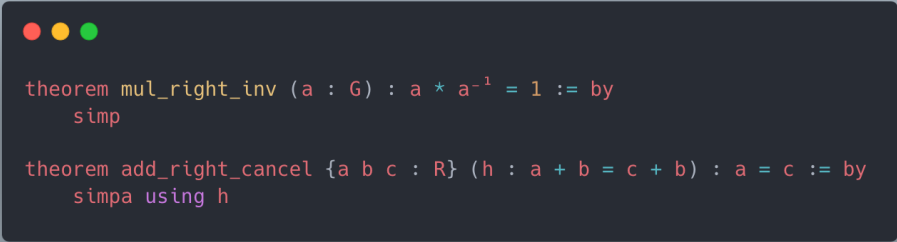
Repository	Total	LeanAgent	Setup 1	Setup 2	Setup 3	Setup 4	Setup 5	Setup 6	Setup 7
PFR	37	1	0	0	0	0	0	0	0
MIL	29	21	16	12	16	16	14	17	14
SciLean	294	27	25	22	20	25	25	26	23

1039 A.5 FURTHER *sorry* THEOREM PROVING DISCUSSION

1042 In addition to comparing LeanAgent and ReProver, we conduct an ablation study between LeanA-
 1043 gent and the seven variants discussed in Sec. 4.3 regarding the original curriculum. The detailed
 1044 *sorry* theorem proving comparison, which focuses on the repositories compared in Sec. 4.2, is in
 1045 Table 9. Note that when using the Merge All strategy, only *sorry* theorems from the new repository
 1046 are proven during each iteration of lifelong learning. We devote the rest of this section to the detailed
 1047 comparison of *sorry* theorems that these setups can prove.

1048 **Mathematics in Lean Source.** We notice a progression in LeanAgent’s understanding of the Math-
 1049 ematics in Lean Source repository. During lifelong learning, LeanAgent demonstrates a strong grasp
 1050 of fundamental algebraic structures and basic mathematical operations:

1051 a) Group and Ring Theory: LeanAgent proves theorems about basic algebraic structures. For in-
 1052 stance, `MyGroup.mul_right_inv` shows that multiplying an element by its inverse yields the iden-
 1053 tity and `MyRing.add_right_cancel` demonstrates the cancellation property in ring addi-
 1054 tion.
 1055



```

1063 theorem mul_right_inv (a : G) : a * a⁻¹ = 1 := by
1064   simp
1065
1066 theorem add_right_cancel {a b c : R} (h : a + b = c + b) : a = c := by
1067   simp using h
  
```

1077 b) Elementary Number Theory: LeanAgent handles fundamental arithmetic properties, in-
 1078 cluding `MyRing.zero_mul`, which proves that zero multiplied by any number is zero, and
 1079 `MyRing.neg_neg`, which shows that the negative of a negative number is the original number.

1080
 1081
 1082
 1083
 1084
 1085
 1086
 1087
 1088
 1089
 1090
 1091
 1092
 1093
 1094
 1095
 1096
 1097
 1098
 1099
 1100
 1101
 1102
 1103
 1104
 1105
 1106
 1107
 1108
 1109
 1110
 1111
 1112
 1113
 1114
 1115
 1116
 1117
 1118
 1119
 1120
 1121
 1122
 1123
 1124
 1125
 1126
 1127
 1128
 1129
 1130
 1131
 1132
 1133

```

theorem zero_mul (a : R) : 0 * a = 0 := by
  rw [MulZeroClass.zero_mul]

theorem neg_neg (a : R) : - -a = a := by
  simp
  
```

c) Order Theory: LeanAgent grasps order theory, as evidenced by `absorb1`, which proves that the infimum of x and the supremum of x and y is always equal to x , and `absorb2`, which demonstrates that the supremum of x and the infimum of x and y is always equal to x .

```

theorem absorb1 : x ⊓ (x ⊔ y) = x := by
  simp

theorem absorb2 : x ⊔ x ⊓ y = x := by
  simp
  
```

d) Rudimentary Real Analysis: LeanAgent demonstrates an early understanding of properties related to real numbers and absolute values, as shown by `C03S05.MyAbs.abs_add`, which proves the triangle inequality for real numbers.

1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187

```
theorem abs_add (x y : ℝ) : |x + y| ≤ |x| + |y| := by
  apply abs_add_le
```

10/14 of these proven *sorry* theorems from Mathematics in Lean Source during the lifelong learning process are from the exercise file for proving identities about algebraic structures. This indicates that LeanAgent starts its understanding of mathematical concepts from the basics.

Crucially, by the end of the lifelong learning process, LeanAgent exhibits significant growth in its mathematical reasoning abilities:

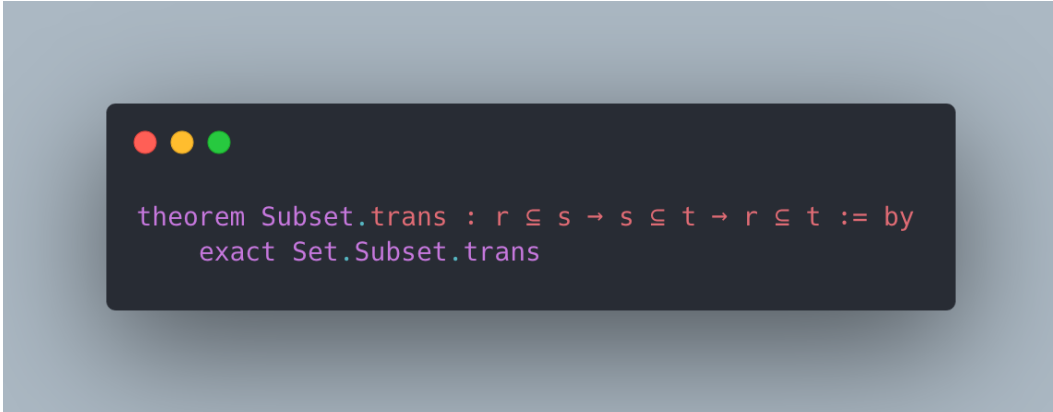
a) **Quantifier Manipulation:** LeanAgent exhibits advanced logical reasoning by managing multiple quantifiers and implications, as evidenced by `C03S01.my_lemma3`, which proves a complex statement involving bounds and absolute values with multiple quantifiers and conditions, and `C03S05.MyAbs.abs_lt`, which establishes that the absolute value of x being less than y is equivalent to $-y < x \wedge x < y$.

```
theorem my_lemma3 :
  ∀ {x y ε : ℝ}, 0 < ε → ε ≤ 1 → |x| < ε → |y| < ε → |x * y| < ε := by
  apply C03S01.my_lemma

theorem abs_lt : |x| < y ↔ -y < x ∧ x < y := by
  cases x
  exact abs_lt
```

b) **Set Theory and Relations:** LeanAgent understands abstract set-theoretic concepts, as shown by `C03S01.Subset.trans`, which proves that subset relations are transitive.

1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241



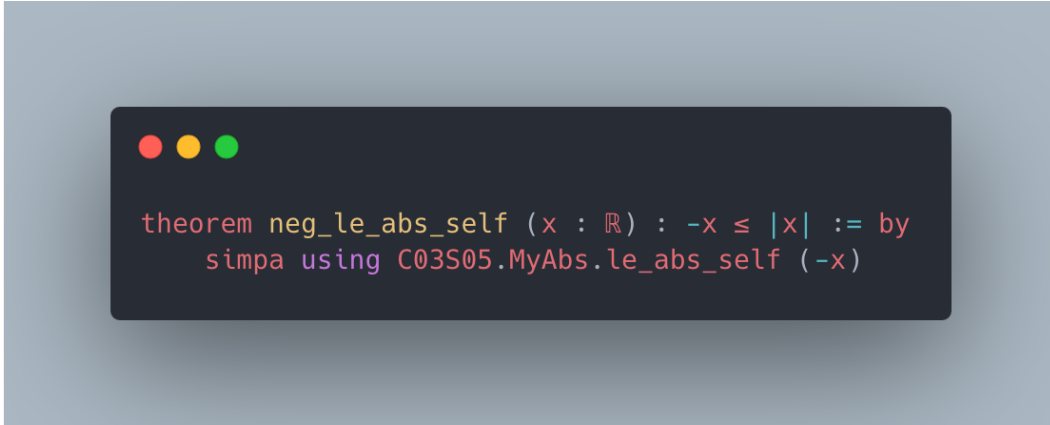
Now, only 2/7 *sorry* theorems from Mathematics in Lean Source are from the exercise file for proving identities about algebraic structures. This suggests that lifelong learning allowed LeanAgent to transition to gaining a stronger understanding of how to use premises for more complicated proofs.

We gain some insights from comparing the performance of LeanAgent over time on Mathematics in Lean Source to other setups. For example, the fact that ReProver can handle harder theorems out of the box, such as `C03S01.my_lemma3`, but fewer theorems overall suggests that it has a broader knowledge base initially but loses performance from a lack of adaptability. Furthermore, Setup 5 proves the same *sorry* theorems as LeanAgent does during lifelong learning. This suggests that pure curriculum learning without EWC or the Merge All strategy emphasizes understanding easier concepts earlier, showing a learning pattern similar to human progression. This mimics the insights gained from the lifelong learning analysis. However, Setups 3 and 7 (curriculum with EWC and/or Merge All) demonstrate some knowledge plasticity, proving harder theorems during lifelong learning, such as `C03S01.Subset.trans` in Setup 3. However, this comes at the cost of an understanding of basic theorems, showing catastrophic forgetting. For example, Setups 3 and 7 could not prove the trivial theorems `MyGroup.mul_right_inv` and `MyRing.zero_mul`, respectively, during lifelong learning, whereas LeanAgent could. This again aligns with the insights from the composite score from our previous analysis.

By the end of lifelong learning, the *sorry* theorems that LeanAgent prove from Mathematics in Lean Source are a superset of those that the other setups prove. This shows that LeanAgent's lifelong learning setup provides continuously improving capabilities to reason about more advanced premises and proofs than other setups. For example, LeanAgent is the only system, except for Setup 6, which can prove theorem `C03S05.MyAbs.abs_lt`. LeanAgent achieved this through its deep understanding of the available premises, such as `abs_lt` with a statement similar to `C03S05.MyAbs.abs_lt`.

An interesting case study can be found in the dichotomy between the theorems `C03S05.MyAbs.neg_le_abs_self` and `C03S05.MyAbs.le_abs_self`. LeanAgent can prove `C03S05.MyAbs.neg_le_abs_self` by referencing `C03S05.MyAbs.le_abs_self`, which is still unproven at that point:

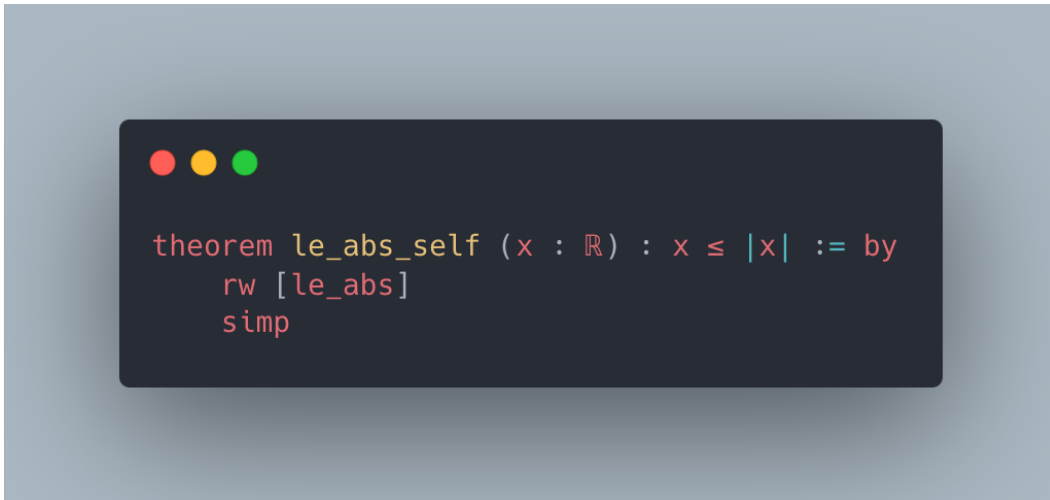
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259



```
theorem neg_le_abs_self (x : ℝ) : -x ≤ |x| := by
  simp using C03S05.MyAbs.le_abs_self (-x)
```

1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283

At the end of lifelong learning, LeanAgent can prove `C03S05.MyAbs.le_abs_self`:



```
theorem le_abs_self (x : ℝ) : x ≤ |x| := by
  rw [le_abs]
  simp
```

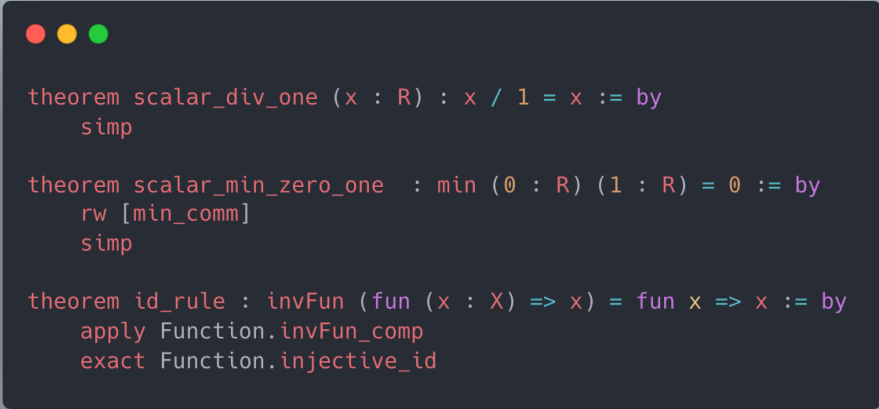
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295

It achieves this through its deeper understanding of the `le_abs` premise, which provides conditions for when an element is less than or equal to the absolute value of another. This suggests that LeanAgent begins by using existing knowledge where possible before trying to deeply understand why existing facts are reasonable.

SciLean. We examine the *sorry* theorems from SciLean that LeanAgent proved to gain some further key insights about its performance. During the lifelong learning process, LeanAgent demonstrated proficiency in a wide range of mathematical concepts from SciLean. These theorems primarily focus on:

a) Fundamental Algebraic Structures: LeanAgent proves basic algebraic operations and properties, such as `SciLean.scalar_div_one`, which proves that dividing any number by one yields the same number, `SciLean.scalar_min_zero_one`, which demonstrates the minimum value between 0 and 1 is 0, and `Function.invFun.id_rule`, which proves that the inverse of the identity function is the identity function itself.

1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318



```

theorem scalar_div_one (x : R) : x / 1 = x := by
  simp

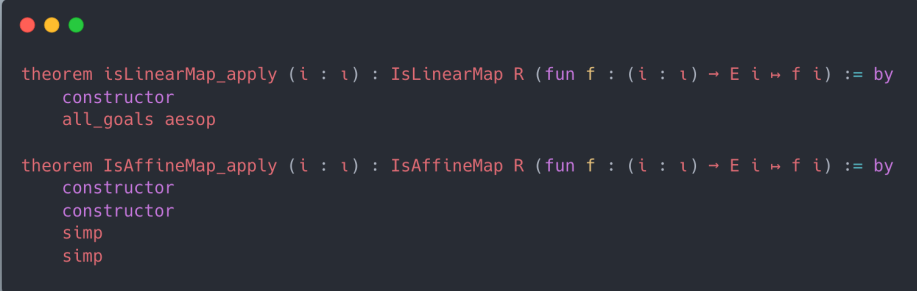
theorem scalar_min_zero_one : min (0 : R) (1 : R) = 0 := by
  rw [min_comm]
  simp

theorem id_rule : invFun (fun (x : X) => x) = fun x => x := by
  apply Function.invFun_comp
  exact Function.injective_id

```

1319 b) Linear and Affine Maps: LeanAgent handles basic properties of linear and affine maps effectively,
1320 recognizing their structure in `IsLinearMap.isLinearMap_apply`, which proves the linearity
1321 of function applications, and `IsAffineMap.IsAffineMap_apply`, which demonstrates the
1322 affine property of function applications.

1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343



```

theorem isLinearMap_apply (i : ι) : IsLinearMap R (fun f : (i : ι) → E i → f i) := by
  constructor
  all_goals aesop

theorem IsAffineMap_apply (i : ι) : IsAffineMap R (fun f : (i : ι) → E i → f i) := by
  constructor
  constructor
  simp
  simp

```

1344 c) Measure Theory Basics: LeanAgent starts grasping measure theory concepts, exem-
1345 plified by `SciLean.ite_pull_measureOf`, which handles conditional measure selec-
1346 tion between two measures based on a proposition, `SciLean.Measure.prod_volume`,
1347 which proves that the product of two volume measures is the volume measure itself, and
1348 `SciLean.ite_pull_enreal_toReal`, which proves that conditionally pulling out an ex-
1349 tended non-negative real and converting it to a real is equivalent to converting the individual com-
ponents first.

1350
 1351
 1352
 1353
 1354
 1355
 1356
 1357
 1358
 1359
 1360
 1361
 1362
 1363
 1364
 1365
 1366
 1367
 1368
 1369
 1370
 1371
 1372
 1373
 1374
 1375
 1376
 1377
 1378
 1379
 1380
 1381
 1382
 1383
 1384
 1385
 1386
 1387
 1388
 1389
 1390
 1391
 1392
 1393
 1394
 1395
 1396
 1397
 1398
 1399
 1400
 1401
 1402
 1403

```

theorem ite_pull_measureOf {X} [MeasurableSpace X] (c : Prop) [Decidable c]
  (μ v : Measure X) (A : Set X) :
  (if c then μ else v) A
  =
  (if c then μ A else v A) := by
  split_ifs <;> rfl

theorem Measure.prod_volume {X Y} [MeasureSpace X] [MeasureSpace Y] :
  (Measure.prod (volume : Measure X) (volume : Measure Y)) = volume := by
  rfl

theorem ite_pull_enreal_toReal (c : Prop) [Decidable c] (x y : ENNReal) :
  (if c then x else y).toReal
  =
  (if c then x.toReal else y.toReal) := by
  split_ifs <;> rfl

```

d) Floating-Point Operations: LeanAgent demonstrates an early understanding of floating-point representations and their correspondence to real numbers, shown by `SciLean.re_float`, proving that a floating-point number's real-like part is itself.

```

theorem re_float (a : Float) : RCLike.re a = a := by
  exact RCLike.re_eq_self_of_le le_rfl

```

The proofs during this phase are characteristically concise, often using basic tactics like `simp`, `rfl`, or `aesop` that do not use premises. This suggests that LeanAgent recognizes these theorems are straightforward enough to prove without the complex retrieval of premises.

Crucially, by the end of the lifelong learning process, LeanAgent exhibits significant growth in its mathematical reasoning abilities on SciLean, just as it did with Mathematics in Lean Source:

a) Advanced Function Spaces: LeanAgent masters concepts in advanced function spaces, such as `SciLean.ContCDiffMapFD_eta`, which demonstrates the eta reduction property for continuously differentiable maps over finite dimensions.

1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415

```

theorem ContCDiffMapFD_eta (f : X →FD[K,n] Y) : (fun x →FD[K,n] f x) = f := by
  simp only [DFunLike.ext_iff]
  aesop

```

1416
1417
1418
1419
1420
1421
1422
1423

b) **Sophisticated Bijections:** LeanAgent grows in its understanding of product spaces and bijections, proving theorems such as `Function.Bijective.Prod.mk.arg_fst_snd.Bijective_rule_simple'`, which proves the bijectivity of a function that swaps elements in a product space and `Function.Bijective.Equiv.invFun.arg_a0.Bijective_rule`, which proves that the composition of a bijection and its inverse remains bijective. Crucially, understanding these theorems might seem simple, but it demonstrates LeanAgent's understanding of abstract algebraic thinking.

1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441

```

theorem Prod.mk.arg_fst_snd.Bijective_rule_simple'
  : Bijective (fun xy : X×Y => (xy.2, xy.1))
  := by
  constructor <=> intro h
  all_goals aesop

theorem Equiv.invFun.arg_a0.Bijective_rule (f : Y = Z) (g : X → Z) (hf : Bijective g)
  : Bijective (fun x => f.invFun (g x)) := by
  convert hf
  simp [hf]
  exact f.symm.bijective.comp hf

```

1442
1443
1444
1445
1446

c) **Abstract Algebraic Structures:** LeanAgent proves further abstract algebraic properties, including `SciLean.CDifferentiable.id_rule`, which proves that the identity function is continuously differentiable.

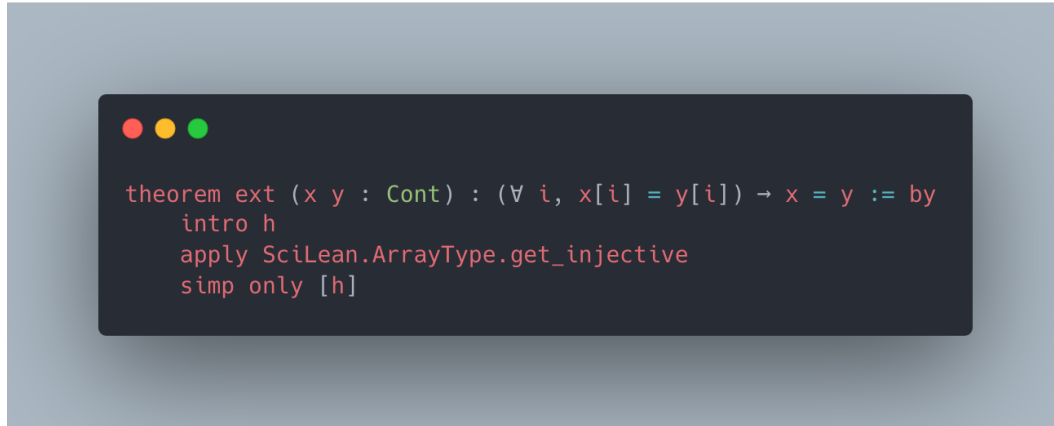
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457

```

theorem CDifferentiable.id_rule : CDifferentiable K (fun x : X => x) := by
  intro x
  unfold SciLean.CDifferentiableAt
  tauto

```

1458 d) Data Structures in Mathematics: LeanAgent understands array types in a mathematical context,
 1459 proving theorems such as `SciLean.ArrayType.ext`, which proves that two arrays are equal if
 1460 their elements are equal at all indices.
 1461



1462
 1463
 1464
 1465
 1466
 1467
 1468
 1469
 1470
 1471
 1472
 1473
 1474
 1475
 1476
 1477
 1478 It is impressive that LeanAgent could understand a traditionally computer-science-oriented data
 1479 structure from a mathematical lens, something that some other setups could not understand.

1480 The proofs at this stage are more sophisticated, involving multiple steps and combining various
 1481 mathematical concepts. This indicates a deeper understanding and ability to connect different areas
 1482 of mathematics.

1483 The progression from basic algebraic structures to advanced function spaces and data structures (like
 1484 array types) shows that LeanAgent is bridging the gap between pure mathematical concepts and their
 1485 applications in computational mathematics. Furthermore, the progression from basic integral man-
 1486 ipulations to advanced function spaces indicates that LeanAgent is improving its premise selection
 1487 over time. It learns to identify and apply more sophisticated mathematical structures and theorems
 1488 as premises. This progression in the complexity of mathematical concepts understood again mirrors
 1489 the typical progression in mathematical education.

1490 By the end of lifelong learning, the *sorry* theorems that LeanAgent proves from SciLean are almost
 1491 entirely a superset of those that the other setups prove. This corroborates our previous assertion from
 1492 our analysis of the *sorry* theorems from Mathematics in Lean Source that LeanAgent’s lifelong
 1493 learning setup provides it with continuously improving capabilities to reason about premises and
 1494 proofs that outperform other setups.

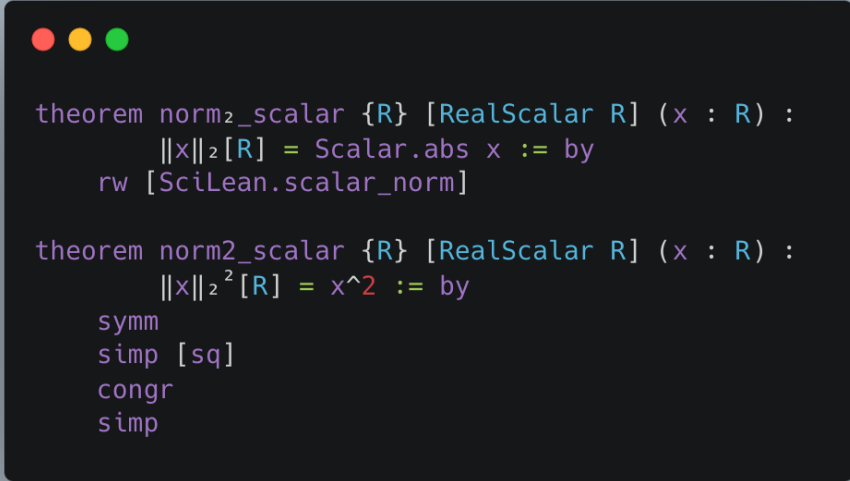
1495 Crucially, LeanAgent could prove `re_float` during lifelong learning while no other setup could.
 1496 This indicates that LeanAgent’s more measured and stable approach allowed it to understand
 1497 floating-point representations and their relation to reals. At the same time, other setups prioritized
 1498 this less with their reduced stability. This may also suggest that continuous improvement allowed
 1499 LeanAgent to understand new and unique concepts from new repositories.

1500 We gain some interesting insights from comparing the performance of LeanAgent over time on
 1501 SciLean to other setups. For example, the fact that ReProver could not prove the trivial theorem
 1502 `SciLean.ite_pull_enreal_toReal` while LeanAgent could, even during lifelong learning,
 1503 suggests that this baseline lacks an understanding of foundational concepts. LeanAgent has a better
 1504 understanding of these concepts from lifelong learning. This is due to the increased stability of
 1505 LeanAgent and improvement from learning a new task, as shown in the lifelong learning metric
 1506 analysis in Sec. 4.3.

1507 Furthermore, an intriguing observation is that Setup 3 could prove
 1508 `Function.Bijective.Prod.mk_arg_fst_snd.Bijective_rule_simple'` during
 1509 lifelong learning. However, as mentioned above, LeanAgent could only prove this theorem at the
 1510 end of lifelong learning. This suggests that Setup 3 is more plastic during lifelong learning, while
 1511 LeanAgent remains more stable. This corroborates the analysis from the lifelong learning metrics.
 However, this again comes at the cost of understanding basic theorems. For example, Setup 3

cannot prove the trivial measure theory theorem `SciLean.Measure.prod_volume`, whereas LeanAgent could during lifelong learning, again suggesting that it favors plasticity over stability.

An interesting case is that LeanAgent can prove `SciLean.norm2_scalar` during lifelong learning but not `SciLean.norm2_scalar`. Conversely, Setup 2 proved `SciLean.norm2_scalar` but not `SciLean.norm2_scalar`.



```

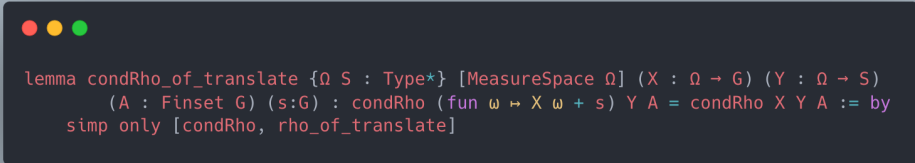
theorem norm2_scalar {R} [RealScalar R] (x : R) :
  ||x||2[R] = Scalar.abs x := by
  rw [SciLean.scalar_norm]

theorem norm2_scalar {R} [RealScalar R] (x : R) :
  ||x||22[R] = x2 := by
  symm
  simp [sq]
  congr
  simp

```

Setup 2 uses the `sq` premise from `mathlib4`, which states that the square of an element is the same as multiplying that element by itself, while LeanAgent used the `SciLean.scalar_norm` premise from `SciLean`, which states that the 2-norm of a real scalar is equal to the absolute value of that scalar. This suggests that LeanAgent prefers to use new premises if possible rather than simplistic pre-trained ones. This also makes sense since Setup 2 uses popularity order, which generally provides poor stability and plasticity.

PFR. Crucially, LeanAgent is the only setup to prove a *sorry* theorem from PFR. We can prove `condRho_of_translate`, which states a form of a translation invariance property of randomness measures.



```

lemma condRho_of_translate {Ω S : Type*} [MeasureSpace Ω] (X : Ω → G) (Y : Ω → S)
  (A : Finset G) (s : G) : condRho (fun ω ↦ X ω + s) Y A = condRho X Y A := by
  simp only [condRho, rho_of_translate]

```

LeanAgent suggests that the proof is straightforward after expanding definitions of `condRho` and considering how `rho`, the randomness measure, behaves under translation (as captured by the `rho_of_translate` lemma). The fact that LeanAgent could trivially identify such a short proof using existing premises while the maintainers of the PFR repository did not suggests the power of

our approach. This is especially powerful when considering the cutting-edge nature of the content of the PFR repository. This suggests that by understanding the foundations of the PFR lemmas using its improved stability, LeanAgent was able to grasp an understanding of some basic definitions.

However, LeanAgent and other setups could not prove any PFR theorems after lifelong learning. This suggests that LeanAgent requires more training time or data to further strengthen its knowledge in this new area of mathematics.

Alternate PFR Commit. We also analyze whether LeanAgent can generalize to a different commit of a repository in the curriculum. We choose commit 861715b9bf9482d2442760169cb2a3ff54091f75, because PFR/RhoFunctional.lean, the file from which we proved `condRho_of_translate` in the newer commit, did not exist in the old commit. This allowed the *sorry* theorems in the old commit to be more distinct. Impressively, it proved two theorems, including the theorem `multiDist_copy` about the equality of distributions when copying random variables across different measure spaces that ReProver could not. LeanAgent achieved this with just the tactic `rfl`, indicating the theorem statements are true by reflexivity.

```

lemma multiDist_copy {m:N} {Ω : Fin m → Type*} {Ω' : Fin m → Type*}
  (hΩ : (i : Fin m) → MeasureSpace (Ω i))
  (hΩ' : (i : Fin m) → MeasureSpace (Ω' i)) (X : (i : Fin m) → (Ω i) → G)
  (X' : (i : Fin m) → (Ω' i) → G)
  (hIdent : ∀ i, IdentDistrib (X i) (X' i) (hΩ i).volume (hΩ' i).volume) :
  D[X ; hΩ] = D[X' ; hΩ'] := by
  rfl

lemma multiDist_of_perm {m:N} {Ω: Fin m → Type*}
  (hΩ : (i : Fin m) → MeasureSpace (Ω i))
  (X : (i : Fin m) → (Ω i) → G) (φ : Equiv.Perm (Fin m)) :
  D[X ; hΩ] = D[fun i ↦ X (φ i); fun i ↦ hΩ (φ i)] := by
  rfl

```

This suggests that LeanAgent has a deep enough understanding of PFR, even beyond the commit it learned from, to notice the connection to reflexivity that even humans did not notice at this commit.

This commit also provides an interesting example of LeanAgent’s power. The PFR maintainers used `0 = 1` as a placeholder for some *sorry* theorems, five of which are `multiTau_min_exists`, `multiTau_min_sum_le`, `sub_multiDistance_le`, `sub_condMultiDistance_le`, `sub_condMultiDistance_le'`. Interestingly, LeanAgent is powerful enough to find loopholes and prove these theorems with this proof:

1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673

```

lemma multiTau_min_exists : 0 = 1 := by
  nontriviality
  simp
  apply @zero_ne_one N _
  exact multidist_eq_zero

```

It combines the known fact that $0 \neq 1$ with the placeholder theorem `multidist_eq_zero`, which states $0 = 1$. As such, it proves `False`, allowing it to derive anything, including $0 = 1$. This demonstrates LeanAgent’s deep understanding of logical manipulation within the PFR repository.

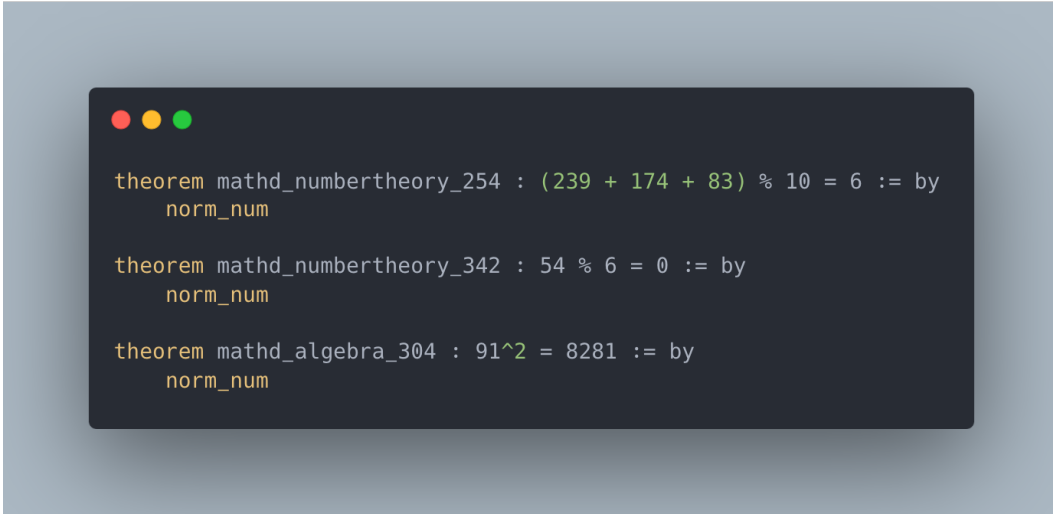
MiniF2F. This repository consists of a validation and test split. Prior work evaluated performance as the number of *sorry* theorems proved and a Pass@k metric on the test split (Yang et al., 2023). However, given that LeanAgent is a framework, not a model, quantitatively comparing it with existing methods using a Pass@k metric is misleading. In line with our existing setups, we do not treat MiniF2F as a benchmark. Instead, we disregard its existing splits and compare LeanAgent’s proven *sorry* theorems with those of ReProver.

LeanAgent can prove 99 theorems, while ReProver can only prove 85. As mentioned in Sec. 4.2, we append MiniF2F to the initial curriculum to demonstrate the use case of formalizing a new repository in parallel with the ones in the starting curriculum. As such, interesting observations can be made by comparing ReProver (starting point) with LeanAgent (ending point). The types of *sorry* theorems LeanAgent can prove demonstrate its increasing proficiency in complex mathematical concepts due to lifelong learning.

ReProver initially demonstrated proficiency in a range of foundational mathematical areas on MiniF2F:

a) **Basic Arithmetic and Number Theory:** ReProver could handle simple arithmetic and modular arithmetic problems, such as `mathd_numbertheory_254`, a theorem about modular arithmetic and basic addition, `mathd_numbertheory_342`, a theorem about basic divisibility, and `mathd_algebra_304`, a theorem about simple exponentiation. These proofs only rely on the `norm_num` tactic, which evaluates arithmetic expressions. This suggests a less sophisticated understanding of mathematics at the start of lifelong learning.

1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727



```
theorem mathd_numbertheory_254 : (239 + 174 + 83) % 10 = 6 := by
  norm_num

theorem mathd_numbertheory_342 : 54 % 6 = 0 := by
  norm_num

theorem mathd_algebra_304 : 91^2 = 8281 := by
  norm_num
```

b) Elementary Algebra: ReProver could solve basic algebraic equations and perform straightforward manipulations, such as `mathd_algebra_141`, which proves a statement about quadratic expressions, `mathd_algebra_329`, which shows a grasp of systems of linear equations, and `mathd_algebra_547`, which proves basic algebraic manipulation with roots.

1728
 1729
 1730
 1731
 1732
 1733
 1734
 1735
 1736
 1737
 1738
 1739
 1740
 1741
 1742
 1743
 1744
 1745
 1746
 1747
 1748
 1749
 1750
 1751
 1752
 1753
 1754
 1755
 1756
 1757
 1758
 1759
 1760
 1761
 1762
 1763
 1764
 1765
 1766
 1767
 1768
 1769
 1770
 1771
 1772
 1773
 1774
 1775
 1776
 1777
 1778
 1779
 1780
 1781

```

theorem mathd_algebra_141
  (a b : ℝ)
  (h1 : (a * b)=180)
  (h2 : 2 * (a + b)=54) :
  (a2 + b2) = 369 := by
  nlinarith

theorem mathd_algebra_329
  (x y : ℝ)
  (h0 : 3 * y = x)
  (h1 : 2 * x + 5 * y = 11) :
  x + y = 4 := by
  linarith

theorem mathd_algebra_547
  (x y : ℝ)
  (h0 : x = 5)
  (h1 : y = 2) :
  Real.sqrt (x ^ 3 - 2 ^ y) = 11 := by
  simp [h0, h1, sq]
  rw [Real.sqrt_eq_iff_sq_eq] <;> norm_num

```

c) Basic Calculus and Analysis: ReProver showed early capabilities in dealing with logarithms and exponentials, including `mathd_algebra_484`, a theorem involving dividing logarithmic expressions.

```

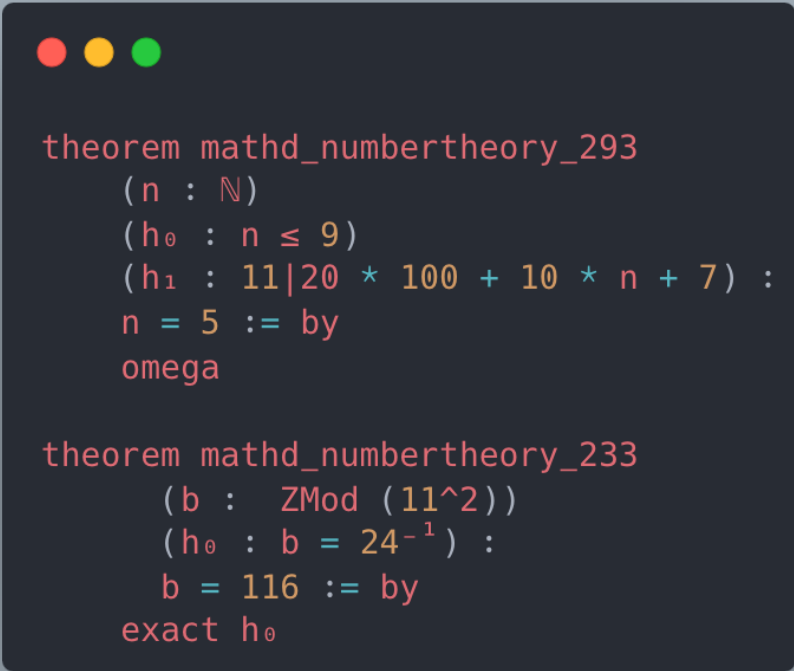
theorem mathd_algebra_484 : Real.log 27 / Real.log 3 = 3 := by
  field_simp
  rw [← Real.log_rpow]
  all_goals norm_num

```

1782 Notably, the proofs at this stage were characteristically concise, often using basic tactics like
 1783 `norm_num`, `linarith`, and `field_simp`. This suggests that ReProver recognized these theo-
 1784 rems as straightforward enough to prove without complex retrieval of premises, similar to its behav-
 1785 ior with previous repositories.

1786 However, by the end of the lifelong learning process, LeanAgent exhibited significant growth in its
 1787 mathematical reasoning abilities on MiniF2F:
 1788

1789 a) Advanced Number Theory: LeanAgent showed a more advanced understanding of number the-
 1790 ory, proving theorems like `mathd_numbertheory_293`, a complex theorem about divisibility
 1791 involving a complex expression and `mathd_numbertheory_233`, a theorem dealing with modu-
 1792 lar arithmetic in $\mathbb{ZMod}(11^2)$.
 1793
 1794
 1795
 1796
 1797
 1798
 1799
 1800
 1801
 1802
 1803
 1804
 1805
 1806
 1807
 1808
 1809
 1810
 1811
 1812
 1813
 1814
 1815
 1816
 1817
 1818
 1819
 1820
 1821
 1822
 1823
 1824
 1825
 1826
 1827
 1828
 1829
 1830
 1831
 1832
 1833



```

theorem mathd_numbertheory_293
  (n : ℕ)
  (h₀ : n ≤ 9)
  (h₁ : 11 | 20 * 100 + 10 * n + 7) :
  n = 5 := by
  omega

theorem mathd_numbertheory_233
  (b : ZMod (11^2))
  (h₀ : b = 24⁻¹) :
  b = 116 := by
  exact h₀
  
```

1834 b) Sophisticated Algebra: LeanAgent showed proficiency in more complex algebraic manipulations.
 1835 Theorems include `mathd_algebra_148`, which involves function definitions and solving for un-
 known coefficients, and `amc12a_2016_p3`, involving a special case of a function.

1836
 1837
 1838
 1839
 1840
 1841
 1842
 1843
 1844
 1845
 1846
 1847
 1848
 1849
 1850
 1851
 1852
 1853
 1854
 1855
 1856
 1857
 1858
 1859
 1860
 1861
 1862
 1863
 1864
 1865
 1866
 1867
 1868
 1869
 1870
 1871
 1872
 1873
 1874
 1875
 1876
 1877
 1878
 1879
 1880
 1881
 1882
 1883
 1884
 1885
 1886
 1887
 1888
 1889

```

theorem mathd_algebra_148
  (c : ℝ)
  (f : ℝ → ℝ)
  (h₀ : ∀ x, f x = c * x^3 - 9 * x + 3)
  (h₁ : f 2 = 9) :
  c = 3 := by
  linarith [h₀ 2]

theorem amc12a_2016_p3 (f : ℝ → ℝ → ℝ)
  (h₀ : ∀ x, ∀ (y) (_ : y ≠ 0), f x y = x - y * Int.floor (x / y)) :
  f (3 / 8) (-2 / 5) = -(1 / 40) := by
  norm_num [h₀]
  field_simp
  norm_cast

```

c) Advanced Calculus and Analysis: LeanAgent demonstrated improved capabilities in handling more complex analytical problems, including `mathd_algebra_270`, a theorem involving function composition and rational expressions.

```

theorem mathd_algebra_270
  (f : ℝ → ℝ)
  (h₀ : ∀ x, x ≠ -2 -> f x = 1 / (x + 2)) :
  f (f 1) = 3/7 := by
  set_option tactic.skipAssignedInstances false in norm_num [h₀]

```

d) Complex Induction: LeanAgent became adept at more advanced induction proofs. An example is `induction_12dvd4expnp1p20`, a theorem about divisibility that requires an induction proof.

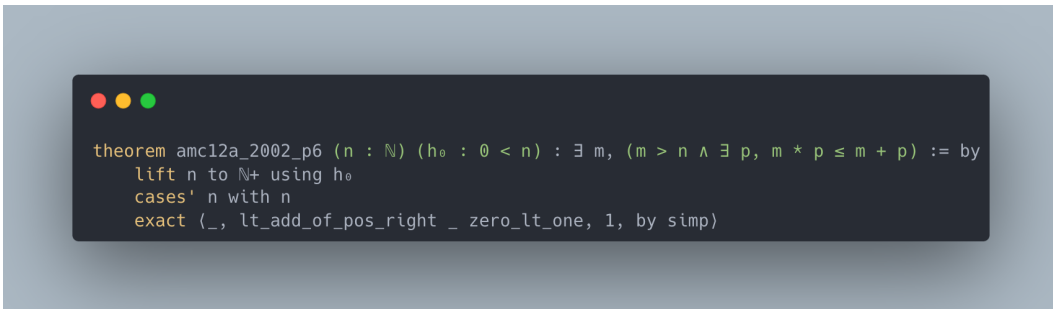
1890
 1891
 1892
 1893
 1894
 1895
 1896
 1897
 1898
 1899
 1900
 1901
 1902
 1903
 1904
 1905
 1906
 1907
 1908
 1909
 1910
 1911
 1912
 1913
 1914
 1915
 1916
 1917
 1918
 1919
 1920
 1921
 1922
 1923
 1924
 1925
 1926
 1927
 1928
 1929
 1930
 1931
 1932
 1933
 1934
 1935
 1936
 1937
 1938
 1939
 1940
 1941
 1942
 1943



```

theorem induction_12dvd4expnp1p20 (n : ℕ) : 12 ∣ 4^(n+1) + 20 := by
  norm_num
  induction' n with n hn
  simp
  omega
  
```

e) Complex Quantifiers and Inequalities: LeanAgent increased its understanding of more complex logical statements, such as `amc12a_2002_p6`, a theorem involving multiple existential quantifiers and inequalities.



```

theorem amc12a_2002_p6 (n : ℕ) (h₀ : 0 < n) : ∃ m, (m > n ∧ ∃ p, m * p ≤ m + p) := by
  lift n to N+ using h₀
  cases' n with n
  exact (_, lt_add_of_pos_right _ zero_lt_one, 1, by simp)
  
```

The proofs at this later stage are more sophisticated, usually involving multiple steps and combining various mathematical concepts or indicating a deeper understanding and ability to connect different areas of mathematics, mirroring the progression observed in Mathematics in Lean Source and SciLean. For example, as shown above, LeanAgent provides a one-line proof to the relatively advanced theorem `mathd_numbertheory_233`. The proof means the hypothesis directly proves the goal. This suggests that LeanAgent has developed a deep understanding of modular arithmetic and can recognize when a given hypothesis is sufficient to prove the goal without additional steps.

Furthermore, as shown above, LeanAgent uses four tactics to prove the `induction_12dvd4expnp1p20` theorem. This demonstrates its ability to handle more complex number theory proofs and use advanced tactics. This again shows that LeanAgent can recognize when its understanding is deep enough to not require complex premise retrieval.

LeanAgent demonstrates a similar understanding of the theorem `amc12a_2002_p6`. Notably, it combines the simple premises `lt_add_of_pos_right`, which describes how an element is less than that element added with a positive one, and `zero_lt_one`, which states that 0 is less than 1, with more advanced tactics like `lift`, `cases`, and `exact` with complex term construction. This demonstrates its ability to reuse foundational understanding for more complex proofs of abstract mathematical concepts, showing its stability.

Importantly, LeanAgent’s performance on MiniF2F showcases its ability to adapt and improve across different mathematical domains. We see this in the progression from ReProver’s basic arithmetic and algebra to LeanAgent’s more advanced number theory, calculus, and abstract algebra. This aligns with the observations from Mathematics in Lean Source and SciLean, further supporting the effectiveness of LeanAgent’s lifelong learning approach in theorem proving across various mathematical repositories.

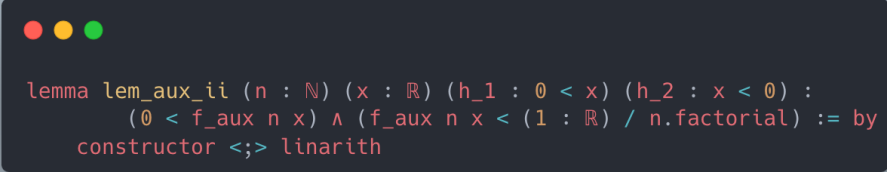
Furthermore, early proofs from ReProver dealt with concrete numbers and simple equations. Later proofs from LeanAgent involved more abstract concepts like equivalence relations and function

1944 properties. LeanAgent gained the capability to handle more complex number theory problems
 1945 involving divisibility under constraints. Moreover, LeanAgent shifted from solving basic linear
 1946 and quadratic equations to analyzing functions and their compositions. Also, early proofs often
 1947 used `norm_num` for straightforward computations. Later proofs employed more varied tactics and
 1948 premises, suggesting a more sophisticated approach to proof construction. This all crucially suggests
 1949 that while existing methods, like ReProver, may be more tailored to simpler computation problems,
 1950 LeanAgent is superior on complex and analytical problems. These are precisely the types of prob-
 1951 lems present in advanced mathematics. This also corroborates LeanAgent’s performance on the
 1952 repositories mentioned previously.

1953 **Formal Book.** We first examine the *sorry* theorems from Formal Book that LeanAgent proved
 1954 during lifelong learning. These theorems centered around:

1955 a) Real Analysis and Inequalities: LeanAgent demonstrates proficiency in real number properties
 1956 and can handle basic inequality reasoning, proving `book.irrational.lem_aux_ii`, which
 1957 involves real analysis and inequalities.
 1958

1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971



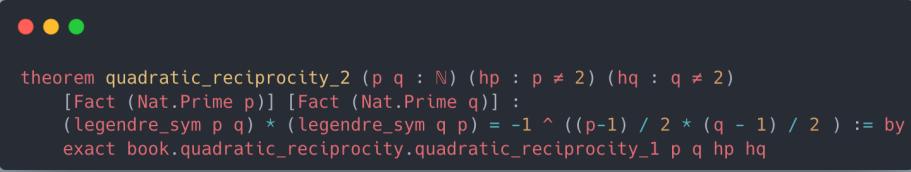
```

lemma lem_aux_ii (n : ℕ) (x : ℝ) (h_1 : 0 < x) (h_2 : x < 0) :
  (0 < f_aux n x) ∧ (f_aux n x < (1 : ℝ) / n.factorial) := by
  constructor <|> linarith
  
```

1972
1973
1974
1975
1976
1977

b) Number Theory: LeanAgent shows capabilities in fundamental number theory concepts, proving
`book.quadratic_reciprocity.quadratic_reciprocity_2`, a key result in quadratic
 reciprocity.

1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989



```

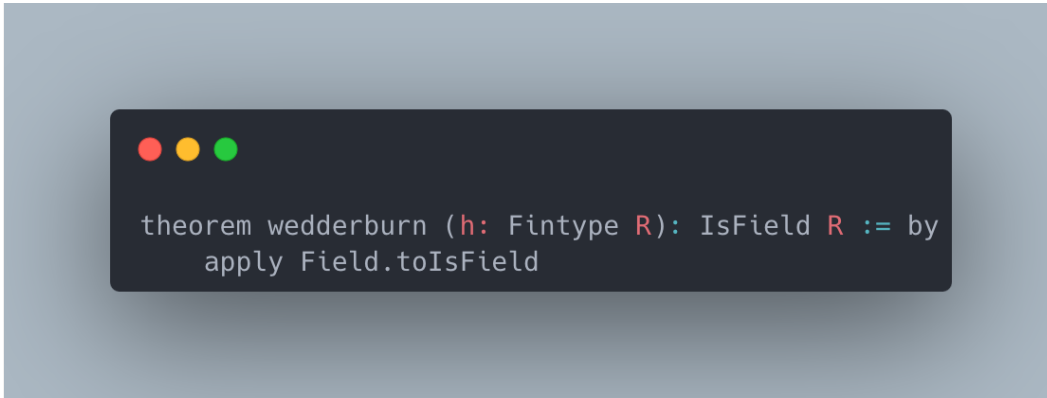
theorem quadratic_reciprocity_2 (p q : ℕ) (hp : p ≠ 2) (hq : q ≠ 2)
  [Fact (Nat.Prime p)] [Fact (Nat.Prime q)] :
  (legendre_sym p q) * (legendre_sym q p) = -1 ^ ((p-1) / 2 * (q - 1) / 2) := by
  exact book.quadratic_reciprocity.quadratic_reciprocity_1 p q hp hq
  
```

1990
1991
1992
1993
1994
1995

Notably, the proof of the first theorem uses no premises, and the proof of the second uses a simple
 statement of quadratic reciprocity. However, by the end of the lifelong learning process, LeanAgent
 exhibits growth in its proving abilities in this repository:

1996 a) Advanced Abstract Algebra: LeanAgent shows significant advancement in proving a key result
 1997 in abstract algebra, `wedderburn` (Wedderburn’s Little Theorem), which is a profound result in
 abstract algebra, stating that every finite division ring is a field.

1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014

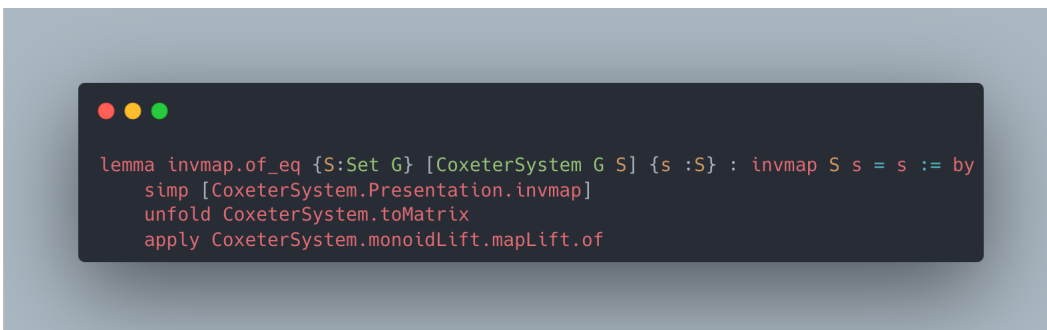


```
theorem wedderburn (h: Fintype R): IsField R := by
  apply Field.toIsField
```

2015 LeanAgent’s proof of the `wedderburn` theorem impressively represents a deep understanding of
2016 algebraic structures. By using the `Field.toIsField` premise, LeanAgent shows that it has
2017 grasped what makes a ring a field and can apply this knowledge efficiently. This requires a com-
2018 prehensive understanding of ring theory and field properties. Impressively, the human proof of
2019 `wedderburn` written after LeanAgent’s proof contains *175 lines*, while LeanAgent’s proof is only
2020 one line. This suggests that LeanAgent has developed a deep, intuitive grasp of abstract algebra.

2021 **Coxeter.** LeanAgent could not prove *sorry* theorems from the Coxeter repository during lifelong
2022 learning. However, by the end of lifelong learning, LeanAgent demonstrates growing proficiency
2023 in more complex algebraic structures, proving the lemma `invmap.of_eq` about Coxeter systems,
2024 again showing the ability to work with advanced concepts in group theory and abstract algebra. This
2025 corroborates the deeper understanding of abstract algebra necessary to prove the `wedderburn`
2026 theorem.

2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044



```
lemma invmap.of_eq {S:Set G} [CoxeterSystem G S] {s :S} : invmap S s = s := by
  simp [CoxeterSystem.Presentation.invmap]
  unfold CoxeterSystem.toMatrix
  apply CoxeterSystem.monoidLift.mapLift.of
```

2045 LeanAgent’s proof of `invmap.of_eq` involves unfolding definitions and applying specific prop-
2046 erties of Coxeter systems. This demonstrates LeanAgent’s growing proficiency in abstract algebra
2047 specific to the new repository it has learned from.

2048 **Hairy Ball Theorem.** Moreover, LeanAgent could not prove *sorry* theorems from the Hairy Ball
2049 Theorem repository during lifelong learning. However, at the end of lifelong learning, LeanAgent
2050 again demonstrates growing proficiency in algebraic topology. It proves `HairyBallDiff`, which
2051 states a key step in the Hairy Ball Theorem, demonstrating an understanding of vector spaces, norms,
and their connections to topological concepts.

2052
 2053
 2054
 2055
 2056
 2057
 2058
 2059
 2060
 2061
 2062
 2063
 2064
 2065
 2066
 2067
 2068
 2069
 2070
 2071
 2072
 2073
 2074
 2075
 2076
 2077
 2078
 2079
 2080
 2081
 2082
 2083
 2084
 2085
 2086
 2087
 2088
 2089
 2090
 2091
 2092
 2093
 2094
 2095
 2096
 2097
 2098
 2099
 2100
 2101
 2102
 2103
 2104
 2105

```

theorem HairyBallDiff : ∃ x, v x = 0 := by
  use 0
  rw [← norm_eq_zero]
  rw [vUnit, norm_zero]
  
```

Crucially, only LeanAgent could prove `invmap.of_eq`, `wedderburn`, and `HairyBallDiff`, demonstrating that it has developed much more advanced theorem-proving capabilities than Setup 3. These proofs show that LeanAgent can work with highly abstract concepts and apply them to specific mathematical objects.

A.6 CURRICULUM LEARNING ANALYSIS

In this section, we aim to answer the following questions: (1) Why does LeanAgent use e^S (S = number of proof steps) as its complexity metric? (2) Why does curriculum learning work in theorem proving? (3) Why does LeanAgent use curriculum learning instead of other lifelong learning methods?

Complexity Measure. There is no universal measure of proof complexity in Lean or other formal systems. One approach, the length-based measure, involves examining the proof length (number of steps or lines) and the size of the proof term in a formal system. While these can indicate verification complexity, they may not fully capture the complexity of discovering a proof (Arana & Stafford, 2023). Moreover, within the NLP literature, many works have related input length to complexity (Zaremba & Sutskever, 2015; Cirik et al., 2016; Spitkovsky et al.; Subramanian et al., 2017; Chang et al., 2021). Starting with shorter sequences and gradually increasing length improves model quality (Li et al., 2024).

These works demonstrate the gains from basing the complexity measure on input length. As such, we consider the equivalent of length in theorem proving to be the number of proof steps. However, we consider a linear scaling of length naive for theorem proving; it doesn't consider the combinatorial explosion of possible proof paths as the length of the proof increases. As such, we choose an exponential scaling. Notably, a key strength of this choice is it is easy to compute and requires no additional hyperparameters to tune.

We now discuss some alternative complexity metrics and why we chose not to use them in LeanAgent. One option is e^B , where B represents the number of different proof paths that could be explored at each step. Formally, B is defined as the average number of child nodes for each non-leaf node in the proof tree. This is sometimes called the branching factor. We refrain from using this complexity metric as computing this becomes computationally expensive for complex proofs. Moreover, another option is to consider the complexity of the theorem statement to determine complexity. For example, this could be measured by the number of unique symbols, the depth of nested expressions, or the number of quantifiers. However, developing a reliable metric for statement complexity that works across various mathematical domains could be challenging. Moreover, LeanAgent focuses on improving proof generation, so using a metric directly related to the proof process (number of steps) aligns better with this goal than statement complexity. Dependency-based complexity, where we order theorems based on their dependency structure within the mathematical library, wasn't used

2106 for multiple reasons. Namely, a theorem might depend on many simple results but still be relatively
2107 easy to prove, or it might depend on a few results but be very challenging. Furthermore, a topic-
2108 based curriculum would be unsuitable because LeanAgent aims to be a general-purpose framework.
2109 A topic-based approach might bias it towards certain mathematical domains. Moreover, a topic-
2110 based approach does not account for theorems spanning multiple mathematical concepts. Another
2111 option is a form of premise-based complexity measure. However, analyzing premise occurrence
2112 frequencies across repositories could be computationally expensive, especially for large repositories
2113 like SciLean.

2114 **Curriculum Learning.** Prior work suggests that curriculum learning guides the learner towards
2115 better local minima in non-convex optimization problems (Bengio et al., 2009). Crucially, theorem
2116 proving involves navigating a highly non-convex optimization landscape, especially for complex
2117 mathematical statements. The space of possible proofs is vast and complex, with many potential
2118 dead ends and suboptimal solutions to parts of a proof. This makes theorem proving an ideal candi-
2119 date for benefitting from curriculum learning. Moreover, curriculum learning has been shown to
2120 have an effect similar to unsupervised pre-training, acting as a form of regularization (Bengio et al.,
2121 2009). For theorem proving, curriculum learning allows LeanAgent to build a foundational under-
2122 standing of mathematics before attempting more complex theorems, naturally leading to continuous
2123 improvement. This avoids suboptimal proof strategies early in training. Furthermore, this leads
2124 to more robust and generalizable proof techniques that work across a broader range of theorems,
2125 explaining LeanAgent’s *sorry* theorem proving performance. Moreover, the ability to act as a regu-
2126 larizer means curriculum learning prevents the model from overfitting to specific types of proofs or
2127 mathematical domains. This allows for continuous generalizability, explaining LeanAgent’s supe-
2128 rior lifelong learning metric scores.

2128 Moreover, other works from the literature show that curriculum learning biases models towards
2129 building constructive internal representations (Cirik et al., 2016). Specifically, it allows the model
2130 to use the knowledge from earlier steps in later predictions. In theorem proving, this allows Lean-
2131 Agent to learn basic proof skills, which then become building blocks for more complex proofs later
2132 on. This corroborates our analysis of LeanAgent’s progression of proof complexity. Moreover, mul-
2133 tiple past works agree that curriculum learning provides larger gains when training data is limited
2134 (Zaremba & Sutskever, 2015; Cirik et al., 2016; Spitkovsky et al.). This is the case in formal theorem
2135 proving, where the number of formalized theorems and proofs is limited. These past works also state
2136 that curriculum learning leads to better generalization, supporting the observations of LeanAgent’s
2137 superior lifelong learning metrics.

2138 This context supports LeanAgent’s *sorry* theorem proving performance and superiority on lifelong
2139 learning metrics.

2140 **Comparison with Other Lifelong Learning Methods.** Many other lifelong learning methods ex-
2141 ist, such as those mentioned in Sec. 2. However, we chose curriculum learning for the following
2142 reasons. Regularization methods, like EWC, slow down learning on important parameters. How-
2143 ever, the importance of parameters can change as the theorem complexity increases. This helps
2144 explain the lower *sorry* theorem proving performance of EWC methods and their performance on
2145 lifelong learning metrics. Moreover, memory-based techniques store examples from previous tasks
2146 to prevent forgetting. However, this can greatly affect these methods’ balance of stability and plas-
2147 ticity. This can be seen in the lifelong learning metrics of Merge All setups, including the negative
2148 IP values. Knowledge distillation requires a separate teacher model, but curriculum learning is more
2149 efficient as it provides the path to knowledge accumulation in a single model. Since LeanAgent is a
2150 framework, not a model, we refrain from using dynamic architecture adjustment to keep LeanAgent
2151 general to many LLM architectures. Moreover, recent work selectively updates parameters with
2152 the largest momentum magnitudes and uses selective reinitialization to maintain plasticity. How-
2153 ever, these methods often focus on balancing performance across distinct tasks. In theorem proving,
2154 where tasks form a spectrum of increasing complexity, curriculum learning provides a more struc-
2155 tured approach to knowledge accumulation.

2156
2157
2158
2159